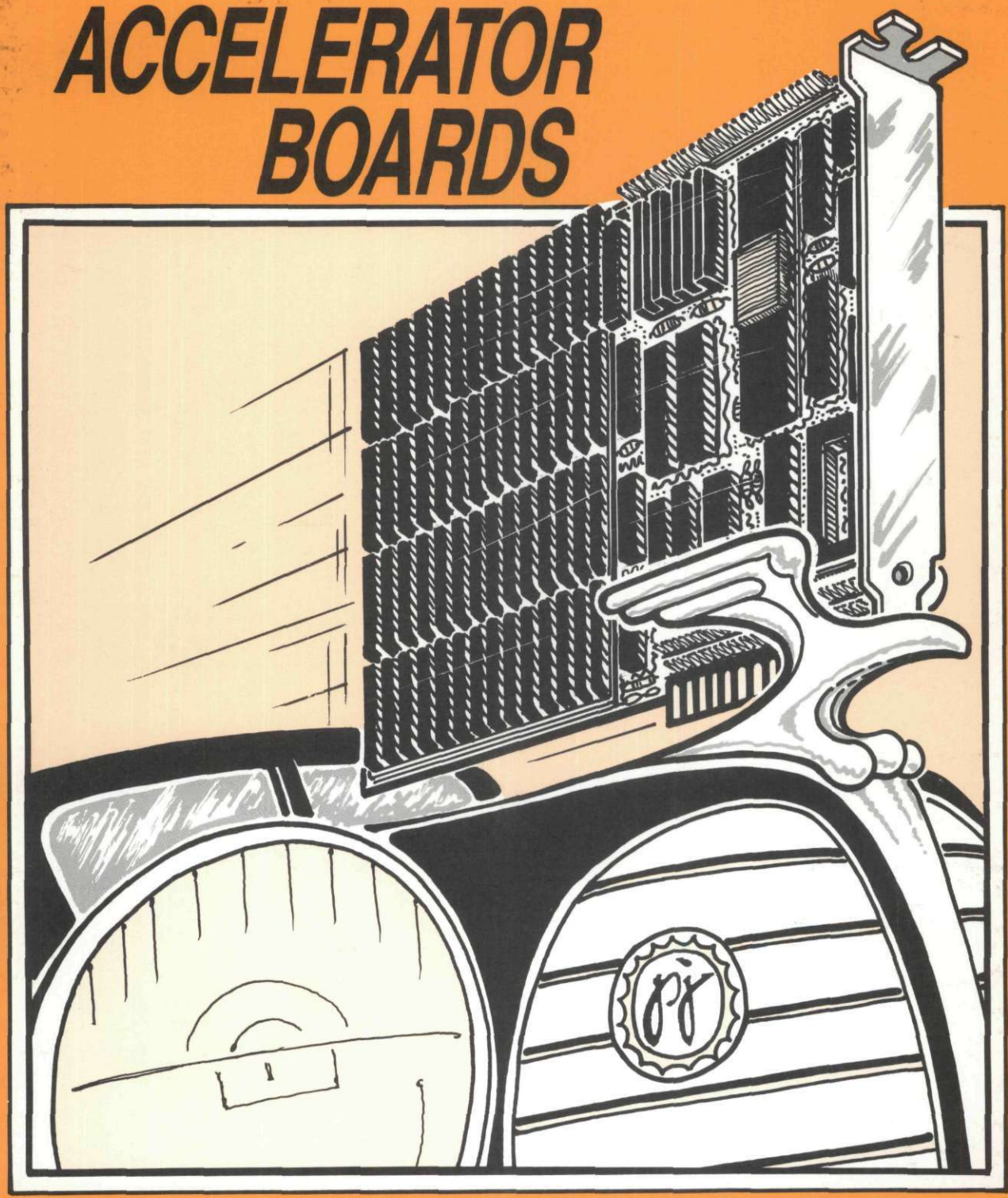
Vol. 4, No. 3 May/June 1986 $4.00

programmer's journal

The Resource Journal for IBM PC Programmers

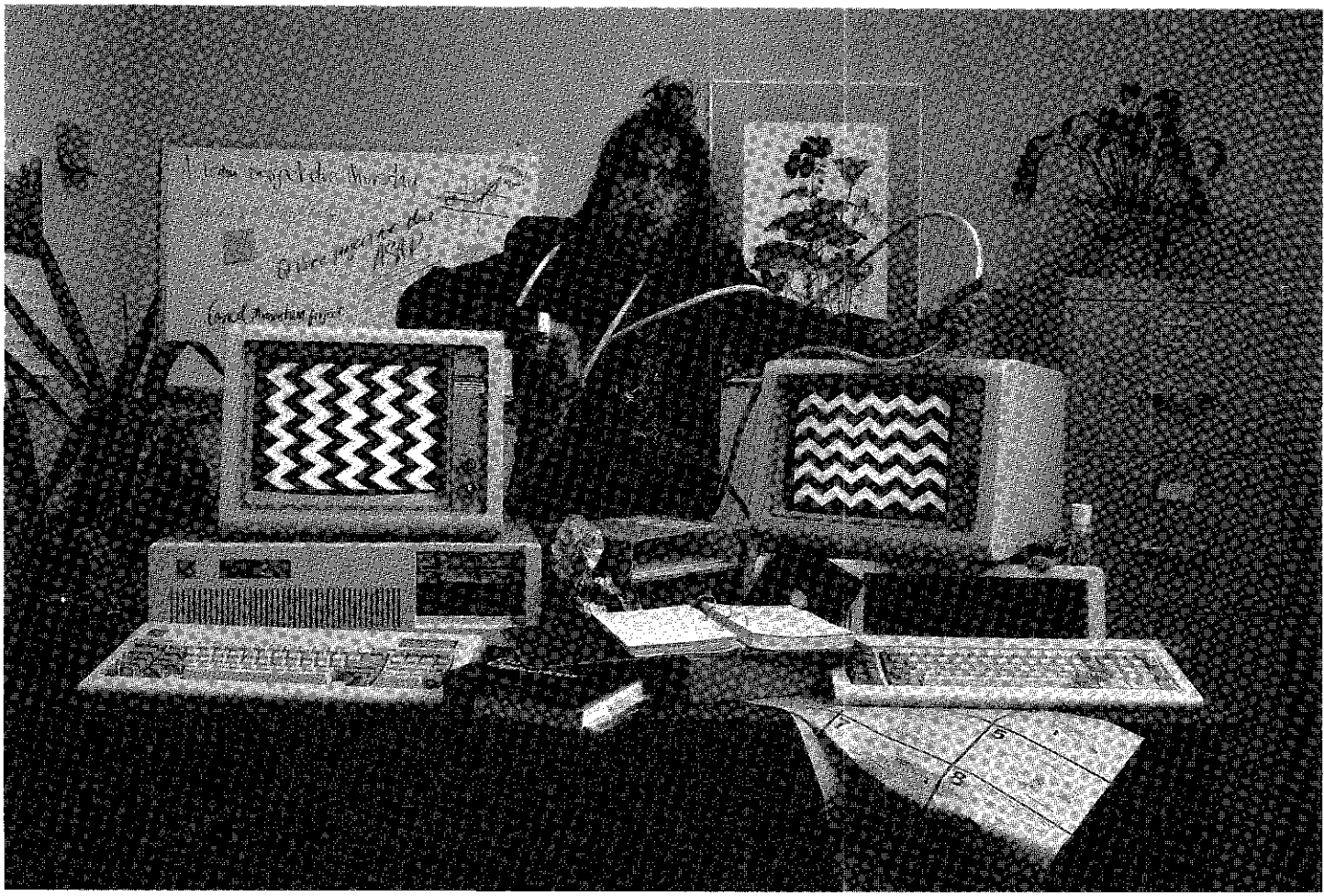
ACCELERATOR BOARDS



BULK RATE
U.S. POSTAGE
PAID
EUGENE, OR 97401
PERMIT NO. 668

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If you can't share files on your network, you're using the wrong file manager.



Be connected. Btrieve.®

Networks can solve problems. But running a single-user file manager can create new ones: Lost updates. Garbled data. Trashed files.

Btrieve® /N offers *safe* multi-user file management that protects your data when sharing files. And eliminates the need to rewrite your application for networking. Btrieve/N set the file management standard for the industry's most popular LANS: IBM's PC Network, Netware, PC Net and Ether-Series. And now Btrieve/N sets the standard for multi-user systems: XENIX, MultiLink Advanced and others.

Fast. Btrieve/N is fast, too. It's written in assembly language especially for the IBM PC. And based on b-tree file indexing with automatic balancing for access speed that won't degrade as your database grows. With Btrieve/N, your applications always run fast. And users don't waste time waiting.

Automatic file recovery. Btrieve/N provides automatic file recovery after a system crash, so accidents and power failures don't turn into database disasters. Your Btrieve data always comes back intact.

Fully-relational data management. SoftCraft's entire family of products gives you a complete, fully-relational database management system. Rtrieve™/N adds report writing capabilities for generating the reports you need. Xtrieve™/N speeds users through database queries with interactive, on-screen menus—no command language or special syntax.

For professional programmers. Btrieve/N is the fast, reliable answer for all your application development. In any development language—BASIC, Pascal, COBOL, C, FORTRAN and APL. With multikey access to records. Unlimited records per file. Duplicate, modifiable, and segmented keys.

With Btrieve/N, you can develop better network applications. And solve problems, not create new ones.

NO ROYALTIES

Suggested retail prices: Btrieve, $245; Btrieve/N, $595; Xtrieve, $195; Xtrieve/N, $395; Rtrieve, $85; Rtrieve/N, $175. Requires PC-DOS or MS-DOS 1.X, 2.X, or 3.X.

Btrieve is a registered trademark and Xtrieve and Rtrieve are trademarks of SoftCraft Inc.

SoftCraft Inc.
P.O. Box 9802 #917
Austin, Texas 78766
(512) 346-8380 Telex 358 200

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

programmer's journal
The Resource Journal for IBM PC Programmers

Volume 4, Number 3 May/June 1986

CONTENTS

NOTE FROM THE EDITOR 2

LETTERS 4

PJ FORUM: ANOTHER RISC MACHINE, THE PAGING GAME, SOFTWARE VIEWPOINT, PROGRAMMER TALENT, AND ALOHA 32032 by FNE 8
Excerpts from DTACK GROUNDED, The Journal of Simple 68000/32081 Systems

REDIRECTIONS: SUN MICROSYSTEMS' NEXT GENERATION PC by Frank D. Greco 18
Why Sun will dominate the workstation market

CONSULTING: REUSABLE MODULES by Paul Barkley 21
Toward a customizable program generator

PASCAL: INT24.PAS, A CRITICAL ERROR HANDLER FOR TURBO PASCAL by Bela Lubkin 27
Trapping and handling DOS Critical Errors within a Turbo program

ASM: MEMORY MODEL INDEPENDENCE IN ASSEMBLER by George Defenbaugh 32
Writing memory model and language independent assembler

ASM: AN APPROACH TO WINDOWING FOR THE ASSEMBLY LANGUAGE PROGRAMMER by C.D. Hawkinson 38
An illustration of one module from an assembler library

C: THE PROPOSED ANSI C LANGUAGE STANDARD, PART 2 by Rex Jaeschke 42
Specifics of the Standard as it comes together

C: MICROSOFT AND LATTICE C 3.0 by William Hunt 47
New versions from Microsoft and Lattice and a getch function you can use with both

REVIEW: ACCELERATOR BOARDS FOR SOFTWARE DEVELOPMENT 51
PJ *examines three 80286 accelerator boards*

MISC 59

PJ BILLBOARD 61

HOUSEKEEPING 62
Funky Errata, Advertiser Index, Peer Review

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

programmer's journal
The Resource Journal for IBM PC Programmers

Publisher: Elizabeth Oakley
Editor: Robert Keller
Managing Editor: Erin O'Connor
Production Manager: Ralph Bentley
Circulation/Promotion: Robert Tappan
Cover: Ralph Bentley
Illustrations: Julia King
Production Assistance: Holly Tracy
Typesetting: Pan Typesetters

Contributing Writers:

Paul Barkley
George Defenbaugh
FNE
Frank D. Greco
C.D. Hawkinson
William Hunt
Robert Hurt
Rex Jaeschke
Bela Lubkin

Founder: Greg Estes

Business and Editorial Offices:
Programmer's Journal
29 West 29th Ave., Suite 5
Eugene, OR 97405
or
P.O. Box 30160
Eugene, OR 97403
(503) 484-2162

Subscription Inquiries:
Programmer's Journal
P.O. Box 3000
Dept. EE
Denville, NJ 07834
(503) 484-2162

Subscriptions: Bi-monthly rates $24 U.S.
($30 in Canada and $40 elsewhere)
2 years $44 U.S.
($55 in Canada and $75 elsewhere)
3 years $64 U.S.
($80 in Canada and $110 elsewhere)

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be published.

Advertising: For ad rates, deadlines, and placement, contact Ralph Bentley or Robin Tappan, (503) 484-2162, or write to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

IBM and Personal Computer are trademarks of International Business Machines; dBASE II and dBASE III are trademarks of Ashton-Tate.

Programmer's Journal (ISSN 0747-5861) is published bi-monthly by Oakley Publishing Company.

POSTMASTER: Send address changes to Programmer's Journal, P.O. Box 3000, Dept EE., Denville, NJ 07834.

© 1986

NOTE FROM THE EDITOR

Peer Review

The best-liked article on the inaugural **Peer Review Card** (*at the back of the last issue of* PJ) is Tom Swan's article *Funky Strings*; second was Neil Rubenking's *Capturing File Handles*. Then, all in a pack, came Paul Barkley's Consulting article, *The Home Office*; *Dtack Grounded*; Rex Jaeschke's column on the *ANSI C Standard*; Frank Greco's *Redirections;* and Steve Baker's *Print Screen Functions*. Congratulations to everybody.

This issue contains another Peer Review Card, with postage prepaid.

The first card asked you to rate the top three articles. For this one, we'd like you to just keep rating until you're satisfied or until you reach the end, whichever comes first.

More I/O

Approximately 8 percent of you sent in the reader survey.

Some results:

What languages do you use?	*What languages would you like to*
1. BASIC 77%	1. C 42%
2. Assembler 70%	2. Assembler 31%
3. Pascal 58%	3. Modula-2 28%
4. C 48%	4. Pascal 23%
5. dBASE 33%	5. Lisp 19%
6. Modula-2 8%	6. Prolog 14%
7. Forth 6%	7. Forth 12%
8. Logo 5%	8. Smalltalk 11%
9. Fortran 5%	9. BASIC 7%
10. Lisp 4%	10. Fortran 7%
11. Prolog 2%	11. dBASE 7%
12. Smalltalk 1%	12. Logo 2%

These two questions may indicate a direction for computer languages. I don't think anyone can foretell the future, though. A couple of years ago Jerry Pournelle predicted that we'd all be writing Modula-2 by now and that the sure way to riches was to go out and spend $40,000 on an AT&T B series computer and then to write software for it. He's still alive and prospering, so no one must be a better fortune-teller.

It's a little strange that the survey returned so many requests to reduce C coverage, considering that C is the language that the most people would like to learn. I guess one could add the Modula-2 and Pascal people together to form one interest group, and that's a large group. Are they the people who dislike C?

What do you like least about PJ?
That's *least.*

It's not a monthly. Too short. Most of the ads are for C people. IBM-specific information. No BASIC language routines. Dtack Grounded's insecure rants on how assembler programmers have more value. Not enough reviews. Not enough ads. Too few articles. Too highly technical. Code listings are too hard to read. More algorithmic approach needed for those lacking a particular compiler (i.e., C). The articles on dBASE and BASIC. High-ego

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

writers, especially assembler fanatics. More C and assembler code. A bit thin for the price. Too d--- much C. No BBS. Lack of beginner's articles. No Modula-2. "C" Cucks. Back issue price is too high. Lack of depth. Too much C. Delete the line drawing fillers. Too much assembly language. Incessant diatribes against Intel CPUs. UNIX will go the way of the pSYSTEM. Non-professional look. Articles on junk languages like Forth, Logo, Prolog, Smalltalk. You are too C oriented . . . What kind of language does NOT check array bounds? . . . This is a professional language? . . . You must be kidding!! . . . Do you even know about COBOL? . . . It ain't elegant, but it does what has to be done. Don't like emphasis on coding over design, specification, and dreams of alternate computational universes. God-awful expensive for scrawny publication. Poorly written articles. Your narrow focus, evidently for the *elite few*. Your preoccupation with the UNIX fad. Articles *very* poorly written. Authors wander aimlessly through topics. Comdex reviews. Too much on languages. Snobbery to IBM. Smart-aleck tone. Cartoons can be done better. Lack of assembler column. Cover page.
That's it for *least*.

Regarding having less C—in the last issue we left the C out of Frank Greco's name.

What you like best about PJ:

Open to opinion, relaxed style. C coverage. Dtack. Frank Greco. Technical orientation. Stays away from the business arena. . . . I hate to be treated like a "DP-professional." Pascal, especially Turbo. It's short. The size is right—I don't have to be a body-builder. Anything pertaining to C. I like very little. Code. You try to talk to individual humans, and not the mythical corporate person. Absence of words like "powerful" and "power users." It's not aimed at beginners. FNE. Unfortunately, Dtack Grounded. Not very much any more. Good quality paper. Ink does not come off in my hands. Reviews and code. High technical level of articles. No "make the most of your modem." Articles offer variety of exposure to the primary languages of the day. Dtack. Pragmatic and useful. FNE!! Few advertisements. Now that Dtack's gone, can you get FNE?

What should we be asking you now?

Should we publish monthly? Would you like to go out for a pizza? Forgiveness for FNE. Is this whole computer thing a fad? Are computers worth all this fuss?

Come on now. You know they are.

—Robert□

Write it once!
MasterFORTH
Portable programming environment

Whether you program on the **Macintosh,** the **IBM PC,** an **Apple II** series, a **CP/M** system, or the **Commodore 64,** your program will run unchanged on all the rest. If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.

MasterFORTH is a state-of-the art implementation of the Forth computer language. Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in macro assembler to make it even faster. MasterFORTH's relocatable utilities, transient definitions, and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint, and trace your way through most programming problems. A string package, file interface, and full screen editor are all standard features. And the optional target compiler lets you optimize your application for virtually any programming environment.

MasterFORTH exactly matches the Forth-83 Standard dialect described in *Mastering Forth* by Anderson and Tracy (Brady, 1984). The standard package includes the book and over 100 pages of supplementary documentation.

MasterFORTH standard package	
Macintosh	$125
IBM PC family (MS DOS 2.1)	125
Apple II family (DOS 3.3)	125
CP/M 2.x (IBM 3740 8")	125
Commodore 64 (with graphics)	100
Extensions	
Software Floating Point	$60
Hardware (8087) Floating Point	60
Graphics (selected systems)	60
Relocator (with utility sources)	60
Target compiler (RAM and ROM)	350
Target Application Generation System (TAGS) - MasterFORTH, Target compiler & relocator	495
Publications	
Printed source listings (each)	$35
Forth-83 International Standard	15

NEW MODEL LIBRARY

List Processor with micro-LISP	$40
Spreadsheet	40
Automatic Structure Charts	40

MICROMOTION
8726 S. Sepulveda Bl. #A17
Los Angeles, CA 90045
(213) 641-4340

Circle Number 13 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LETTERS

"Sub"-Batch Files and Hard Disk Backup

I got several good ideas out of Michael Hoyt's "A Menu-Driven PC Using DISPLAY.COM" in *PJ*, Vol. 3, No. 5.

I had the task of setting up several IBM PC XTs in my office. When several casual users must share computers and a variety of application programs (as is the case where I work), a menu built around a batch file is a good solution. Judging from the batch file listing in Figure 2, our solutions were very similar.

We also took advantage of a freeware program called Extended Batch Language (EBL), which, in effect, adds several additional features to DOS's clumsy offerings. Attractive menu displays with borders, colors, and attributes can easily be placed on the screen. Further, EBL has its own "inkey," which can then be tested with its own "if" statements. EBL also has the ability to push keystrokes onto the keyboard stack so that future questions are already answered (this feature is even smart enough to trick programs which normally flush the buffer).

By the way, in Figure 2, you do not need to test for error level =80, 79, or 75 – 71. We speeded up the operation of the batch file in two other ways as well: (1) We divided it into a main menu batch file which chains to "sub"-batch files for each application and (2) placed the "calls" to these sub-batch files *at the top* of the main batch file (so, begin the batch file with a "goto" around the calls to get to the part which outputs the menu; evidently, DOS searches for labels from the top of the batch file, and this search slows down the operation of the batch file if the label is near the end). Of course, placing the menu.bat into a RAM disk also speeds things up.

As it happened, I wrote a much shorter program (called MY__INKEY.EXE) just to accept the keyboard stroke for the menu selection. This program is invoked right after the menu is laid out on the screen using EBL. This program has three main tasks:

1. Flushes the keyboard buffer, gets a keystroke, and converts the user keystroke to uppercase to simplify testing it later on.
2. Includes a pseudo-screen-blanking feature which clears the screen after a certain time interval and replaces the menu with one line: "SCREEN SAVER ON . . . PRESS ANY KEY TO RETURN TO MENU." (This line then scrolls down the screen, one line per the same time interval).
3. When a menu selection is made, puts "STAND BY" on the screen until the applications program takes over (otherwise, these batch files move so slowly it appears that nothing is happening for several moments).

EBL cannot yet test return codes from programs (Seaware's new version of EBL will), so after MY__INKEY exits, I have to use the ponderous "if errorlevel" business, just as Michael Hoyt had to do.

On the subject of backing up a hard disk—here's how I got around the problem. With applications programs already backed up on their floppy originals, I have various text, worksheet, programming, and miscellaneous files to be backed up. These I have organized into subdirectories. Then I:

1. Installed a 1.2Mb floppy drive, tripling per-disk capacity (a Teac driven by a controller called JDISKETTE by TallTree Systems).
2. Instead of any DOS command, used a program called "JET"(also from TallTree) to copy files onto the floppy diskettes. JET's advantages are (1) speed, (2) the option to copy files only if they have later time-stamps than the ones they are replacing, and (3) the files so copied are, unlike those created by BACKUP, normal DOS files.
3. Put all the hard disk backing up into a secondary menu (called from the main menu). In this menu, my selections are "Backup Text Files," "Backup Worksheets," etc. Usually the backup file will fit on a single 1.2Mb floppy. I do this selective backup, then, only when I have been working with the files directly affected. Usually the whole job amounts to inserting one or two floppies and making a couple menu selections.

Glenn A. Emigh
Whittier, CA 90602

Hard Disks and Operating Systems

Some thoughts regarding two items from the Jan/Feb '86 issue of *Programmer's Journal*: "PJ Forum" and "An Operating System for the People." My understanding is that FNE and Hal Hardenbergh are the same person, and I comment as though they were one. He/they seem, at least, to express similar points of view about the issues I'm responding to.

A fallacy pervading the two pieces is that users, and perhaps the industry, are static targets. Thus, "in late 1985/early 1986 I question just how many average personal computer users will have a hard disk."

In a year or two hard disks will be widely used by his target audience, and then he'll have to modify his design to accommodate the change. If he's fortunate or has given it some forethought, he will be able to do it in a consistent manner. If not, he may end up with kludges, as others have when faced with the same demand from users.

I also believe that personal computer users have tended to reject multiple operating systems for the same machine and have preferred (perhaps grudgingly) adding functionality (and complexity) to the existing dominant operating system. Certainly needing more than one operating system to run the applications one requires is cause for complaint in itself.

All this has ignored non-technical factors which weigh in a product's success, the most significant being marketing. Many technically superb products have failed because of inadequate marketing.

In spite of our disagreement on some of these technical trade-offs, I wish FNE/Hal Hardenbergh great success in his venture into mass market software.

Michael Shook
Bar Harbor, ME 04609

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Get a Grip on Assembly Language.

The Award Winning Visible Computer: 8088.

Assembly language programming is far from easy, but you don't have to be a genius to learn it. Let your PC teach it to you.

The Visible Computer: 8088 is an award-winning mix of book and software for mastering the elusive skills of assembly language.

It's an animated simulation of the PC's microprocessor that takes you inside the 8088 as it executes programs. You'll see how instructions are executed, not just the result.

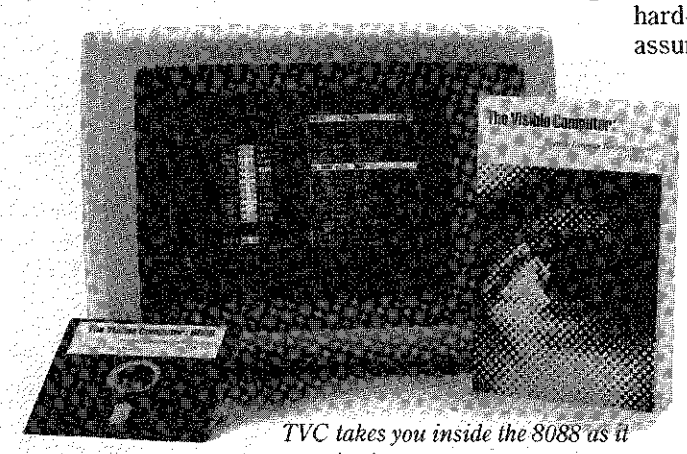
With its intuitive "moving bar" instruction display, memory windows, bulletproof beginner's mode, and more than 30 commands and functions, The Visible Computer is a powerful instrument for inspecting your own programs (or BIOS routines or anything else you can get into memory). You'll be using it as a debugging tool for years to come.

It's a book. The 350 page manual is more than simply instructions on working the simulator – it's a hard-working, assume-nothing tutorial on the fundamental concepts of assembly language. A lot of people think it's the best book on assembly language ever written.

PC Tech Journal "Program of the Month"

"(The Visible Computer) is at once useful and educational, offering an interactive debugger and a self-paced course in 8088 architecture and assembly language... A considerable value."

The Visible Computer: 8088 for IBM PC/XT/AT and true compatibles.



TVC takes you inside the 8088 as it executes programs.

$49.95 Copy Protected

$79.95 Unprotected

If your dealer doesn't have it, order direct: Software Masters, 2714 Finfeather, Bryan, TX 77801 (409) 822-9490. Please enclose $3.00 shipping. Bank cards accepted.

Software Masters™

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Questions for Rex and Proof Positive

I just read Rex Jaeschke's article on the uses and misuses of C in your Vol. 3, No. 1. Rex asserts that C is most certainly suitable for programming operating systems but cautions that C might not be the language of choice for application programs in certain areas—finance, for instance.

However, he does not address the question of whether BASICs should be written in C, as many 68000 BASICs are. I would love to have him address that question in his column. He might also address the question of that 207K TOS + GEM which comes with the Atari 520 ST—isn't that a bit over-large? In the early version of the 520 ST, which did not have the operating system in ROM, the operating system plus BASIC left only 5,000 bytes (!) of user code in a 512K RAM system. The reason is that the operating system and BASIC were written mostly in C or in BCPL, a British near relative of C. What does REX think of this? The size? The incredibly slothful performance?

I am a cad to reveal inside info, but the Norton Editor now being advertised as "lightning fast" (*PJ*, Vol. 4, No. 2, p. 7) is in fact written *entirely* in assembly . . . perhaps I am not the only "fanatical adherent to efficiency?"

Hal W. Hardenbergh
President, Digital Acoustics, Inc.
Santa Ana, CA 92705

We Missed It

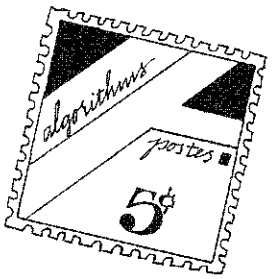
I am writing in reference to the "Report From the Comdex and Usenix Shows," written by Frank D. Greco, which appeared in your November/December 1985 issue in the "Redirections" section.

Under the subsection titled, "The South's Gonna Do It Again," Mr. Greco states that, ". . . Quadram announced Quadgraph, a board that performs graphics on your IBM monochrome screen."

Although Quadram introduced more than a dozen new products at COMDEX/Spring '85 in Atlanta, the Quadgraph was not among them.

The Quadgraph was introduced in the fall of 1984, and the major portion of the advertising on this product was conducted in the same time period. Perhaps this is why Mr. Greco has, ". . . yet to see the board advertised in any computer related publication since Comdex."

Annette Gustafson
Account Executive
Quadram Corporation
Norcross, GA 30093



Prescription for *PJ*

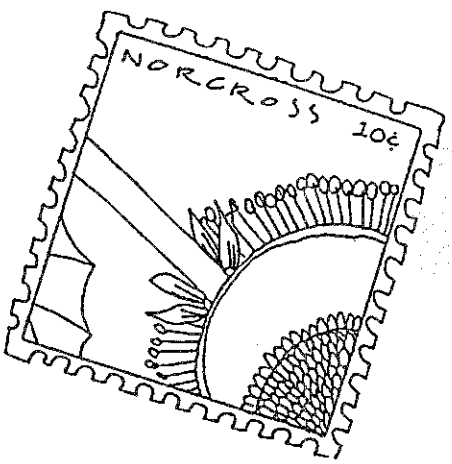
We just received our first issue of *PJ*, and we were a bit disappointed. We would like to see how professional programmers construct those hot new algorithms that have made some of their programs top sellers on the market. We would like to see some algorithms to make windows, animated graphics, resident programs, and fancy screen layouts and processes for multi-tasking.

We are not interested in the language called BASIC; we are interested in the more powerful structured languages such as Pascal, Modula-2, Ada and C, until C becomes obsolete. Yes, structured languages such as Modula-2 and Ada will in the future make C go the way of Fortran.

As soon as the makers of Turbo Pascal and others come out with hot, low-cost Modula-2 and Ada compilers and support them as well as Turbo Pascal was supported, Modula-2 and Ada will sweep the market making C less popular.

We want to see algorithms, algorithms, and more algorithms in *PJ*, written in Pascal, Modula-2, Ada, C and assembly language. We don't care whether the assembly language algorithms are written in 68020, 32032, or 80386, but since this is a journal for IBM PC programmers, we would hope most of the code would be for the 8086 line of microcomputers.

G. Elijah Gwinn
Rochester, WA 98579



80386 vs. 68020

"PJ Forum" in the March/April 1986, issue of *PJ* prompts me to write a few clarifications about the 80386 vs. 68020 comparison. For starters, while it is true that the 386 does not add any registers, it did remove the restrictions on their addressing modes (that is, indexed addressing can use any register, not just DI or SI)! Furthermore, there was a significant addition of two segment registers with no predefined use (286 segment registers are used as follows: SS - stack, CS - code, DS -data, ES - extra. 386 added FS and GS as extras). And, to be honest, I do not feel that the 386 and 68020 are that different in their instruction sets though I sometimes wonder if the 386 may not have more flexible instructions. I'm biased in that opinion, however (I'm currently porting a C compiler to both CPUs).

Lawrence C. Widing
Champaign, IL 61820

Durable Goods

Although I have subscribed to *Programmer's Journal* for less than a year, I thoroughly enjoy the journal and look forward to each issue. Unfortunately, since I run the equivalent of an "information center" for the Maxwell School of Citizenship and Public Affairs (graduate school in social sciences), my office gets a lot of traffic, and my journals have a nasty habit of disappearing for weeks or months on end. On the other hand, the use of high quality paper and readable typeface in the magazine help to ensure its durability, so I have yet to lose an issue. Furthermore, the high quality of the articles (or their quality of representing workable solutions to genuine problems) usually means that during their absence from my office, some user has been putting the journals to work. For example, the dBASE routines have been popular, and the menu system with batch files has been installed on at least a dozen machines that I know of.

Mark D. H. Miller
Syracuse, NY 13244

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Betcha our compiler can beat up your compiler.

Or your money back.

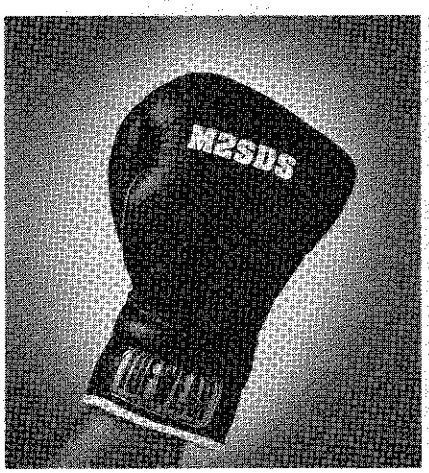
If you program in BASIC, Pascal, Fortran or C, you're using yesterday's technology. We know this statement will start a brawl, but it's true. So, before you start a fight you can't win, take a closer look at the Modula-2 language and the Modula-2 Software Development System (M2SDS) from Interface Technologies. Just compare the features and performance of M2SDS to your system. You'll find a new language and a programming environment that's more flexible, much faster and works on any IBM® PC or 100% compatible with 256 K memory or more.

	M2SDS
COMPILE SPEED (MIN:SEC)	
30 LINES	*0:15.58*
300 LINES	*0:25.48*
EXECUTION SPEED (MIN:SEC)	
SIEVE	*0:13.92*
FIBONACCI	*0:53.49*
30X30 MATRIX (8087)	*0:08.84*
FP OPERATIONS	*0:27.56*
FP OPERATIONS (8087)	*0:01.97*
SYNTAX CHECKING EDITOR	*YES*
MULTIPLE WINDOW EDITING	*YES*
EDITOR FILESIZE LIMIT	*MEMORY SIZE*
COMPILE ERROR CALLS EDITOR	*YES*
LINKER	*YES*
PRODUCES .EXE FILES	*YES*
EXECUTABLE CODE SIZE LIMIT	*DISK SPACE*
DOS ACCESS FROM EDITOR	*YES*
DOS ACCESS FROM PROGRAMS	*YES*
8087 SUPPORT STANDARD	*YES*
COPY-PROTECTED DISK	*NO*
COST WITH 8087 SUPPORT	*$50.88/$80.88*

Source: Software Resources, Inc. Sieve program from BYTE, January 1983. Fibonacci program from Dr. Dobb's Journal, February 1985. Matrix program from BYTE, October, 1982. FP Operations program from BYTE, May 1985. M2SDS with or without 8087 uses 8-byte accuracy. Programs compiled with all checking options on. All tests conducted on a standard IBM-PC/XT with 512K of memory and an 8087 math coprocessor.

*Original or back-up diskette may be sent for trade-in. Diskette will be destroyed immediately upon receipt so that your current compiler license agreement is not violated.

IBM is a registered trademark of International Business Machines Corporation.



Trade In and Trade Up. Just to prove that we're not all brag... we'll send you M2SDS for just $50.88 if you mail us your present compiler or interpreter diskette.* That's $30.00 off the regular price. If within 30 days you're not programming faster than ever, just return the diskette and we'll send you your money back.

Heavyweight Champion SDS-XP. If you're ready to move into light-speed, you need SDS-XP. It has everything M2SDS has with a little "punch" added. Like Extended Libraries, M2MAKE and a Foreign Object Module Importer. SDS-XP offers buyers a stout discount when compared with the cost of buying M2SDS and the additional components as add-ons. For a limited time only, SDS-XP is available for $99.00 with compiler trade-in. That's $150.00 off the advertised price of $249.00.

INTERFACE TECHNOLOGIES

3336 Richmond, Suite 200, Houston, TX 77098

1-800-922-9049

(In Texas, call 713/523-8422) Telex: 322127
In Europe call ITC at: Switzerland 41 (1) 700-3037; Netherlands 31 (20) 106922; U.K. 44 (1) 656-7333

Knock Out Bugs $79.95. Announcing M2DEBUG. A symbolic, compact (only 20K RAM) interactive run-time debugger with features most requested by software programmers and system developers. It comes with a Virtual Resource Overlay Operating Manager (VROOM) that speeds up compile time 50%.

Calling All Compilers. So now that you're wise to the limitations of your system, why not trade it in. You'll soon see that it was smarter to switch than fight. And a pretty safe bet.

Here's my diskette. Rush me:

☐ M2SDS for $50.88 each, plus $7 shipping and handling.

☐ SDS-XP for $99.00 each, plus $7 shipping and handling.

Or, send me:

☐ M2SDS for $80.88 each, plus $7 shipping and handling.

☐ SDS-XP for $249.00 each, plus $7 shipping and handling.

☐ M2DEBUG for $79.95

☐ My check is enclosed.
☐ Apply charges to credit card indicated below:

VISA/MasterCard/American Express (circle one).

Credit Card # _____

Expiration Date _____

Signature _____

Name _____
(please print)

Shipping Address _____

_____

City _____

State/Zip _____

Day Phone _____

Texas residents add 6.125% Sales Tax.
International orders add $30 for shipping/handling.
If paying by check, check or draft must be in U.S. dollars drawn on a U.S. bank.

INTERFACE TECHNOLOGIES CORPORATION
3336 Richmond, Suite 200, Houston, Texas 77098

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Circle 43 on the reader service card

ANOTHER RISC MACHINE, THE PAGING GAME, SOFTWARE VIEWPOINT, PROGRAMMER TALENT, AND ALOHA 32032

Excerpts from DTACK GROUNDED, The Journal of Simple 68000/32081 Systems

by FNE, © 1985 by Digital Acoustics, Inc.

Another RISC Machine:

Electronic News has announced a new 32-bit RISC microprocessor—from ACORN in Great Britain. ACORN is the outfit that won the contract for the "BBC computer" but has lately fallen on lean times and had to be bailed out by Olivetti. ACORN is now a subsidiary of Olivetti. And AT&T owns a big piece of Olivetti. It gets interestinger and interestinger.

The new ACORN RISC machine is CMOS and operates with a 150nsec instruction cycle, and reportedly includes logic to take advantage of the "page mode" of some DRAMs. (Nuts! And we had planned to be the first to publicly break the news of this technique!)

A Momentary Misdirection:

Two and a half years ago, when we and James S. began the design of our VDHR high resolution graphics boards, we knew we had to push a new pixel out every 14.7nsec—roughly the time it takes a beam of light to cross the average bedroom. And we were determined to make that product perform using 2-sided PC boards and TTL technology. Well, we got it to work after six months of effort, but it was NOT easy and both we and James had an, uh, "interesting learning experience!"

When we later undertook the POTBOILER design, which needed a new pixel only every 44.6nsec, James exclaimed, "44.6nsec! Why, that's time enough to play three sets of tennis!"

Well . . . O.K., so that is a small exaggeration. But it is a fact that, with a need for a new pixel every 14.7nsec, we needed to do a 16-bit word fetch every 235.2nsec. At the time the VDHR boards were designed, 120nsec DRAM were both scarce and extremely high-priced. So the VDHR board was designed to use 150nsec DRAM. But—150nsec DRAM had a 260 read cycle time specification—we needed to read a word every 235.2nsec. What to do?

What we did was use the page mode of the DRAM. Meaning that we could read one word, continue to hold #RAS low while bringing #CAS high, changing the least address bit, and bringing #CAS low again to read another word. This way, we were getting 32 bits every #RAS cycle, and we were able to get data fast enough from 150nsec DRAM to push pixels at 14.7nsec intervals. (A redundant set of latches was required to hold the extra word.)

We extended this technique to the next level when we designed the POTBOILER. You will remember that the 7220-based POTBOILER board was designed not for the highest performance—that was the VDHR's forte—but for optimum cost-effectiveness. Having a near-eternity (44.6nsec) to fetch each pixel, we therefore needed only one word every 713.6nsec. But we needed one word for EACH bit-plane in that time-frame. Since the resolution was only (!) 640 x 400, each bit-plane occupied just under 32Kbytes, so four bit-planes could be concentrated into 128K, corresponding to 16 each 64K x 1 DRAMs.

With 713.6nsec to play with, we decided to investigate using page mode to try to fetch four bit-planes during that time so the POTBOILER could have 16 colors (R,G,B,I). Turned out that we could not quite get four words in that time, so the POTBOILER wound up with 3 bit-planes and eight colors. (But it is NOT interlaced! The Amiga's 640 x 400, with 16 colors, IS interlaced. Tsk.)

The 150nsec DRAM access time is the specified time between when #RAS goes low and when the data is available. The fact that #RAS goes low suggests that it has to go high again before the next data access. In general that is true, and it is also true that #RAS must remain high for a specified minimum time, called the RAS precharge time. That's 100nsec for 150nsec DRAM. Add the two together and you get 250nsec minimum read cycle time. Why the spec is 260nsec, 10nsec longer than our simple calculation, is left as an exercise for the student.

Page mode improves on the minimum read cycle time by allowing up to 256 consecutive data fetches while #RAS is held low. The dominant delay is the data access time from #CAS, which is only 100nsec vs. 150nsec from #RAS. Therefore, page mode is 50% faster!

The above applies to most 64K x 1 and 256K x 1 DRAMs.

So we have been shipping one product for two years and another for 1 1/2 years which uses the page mode of 64K x 1 DRAMs to beat the read cycle time restriction (same as the write cycle time spec and generally called the read/write cycle time).

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Microprocessors vs. DRAMs:

At the personal computer end of the computer market, the dominant CPU technology is the microprocessor and the dominant change access memory technology is the DRAM. Ever since the microprocessor and DRAM were invented (both at roughly the same time by a little company called Intel), there has been a race between the micro folk and the DRAM makers. The race has been to see which one could outrun the other.

(Actually, until the last couple of years, the DRAM makers ignored the microprocessors and concentrated on building fast, high-density parts for the mainframe makers. Today the DRAM makers have noticed that the personal computer market uses the odd DRAM or two . . .)

Right now, 150nsec is the vanilla access time and 260nsec is the vanilla read/write cycle time for 256K x 1 DRAMs when used in the ordinary (non-page mode) way. 120nsec DRAMs are available for a small premium and provide a small speed advantage—as in our zero-wait-state B.G.+. Meanwhile, the micro merchants are coming out with advanced microprocessors based on 2 micron CMOS (the 68020) and, imminently, 1.5 micro CMOS as in the promised 12MHz Intel 286. The soon-to-be-announced Fairchild RISC machine is also likely to be based on 2 micron CMOS. (We will ignore the, for now, fantasies being promulgated by Nat Semi of 15MHz 32032s and by Zilog of 25MHz Z80000s.)

These new micros flatly need faster DRAM to run without wait states at their high clock rates. The read/write cycle time of conventional 150 or 120nsec DRAM does not cut the mustard. Now, perhaps you will understand why we bored you with a recap of our graphics controller board designs: Page mode operation of those DRAMs points the way toward better performance in the imminent world of high-speed microprocessors.

Enter Hitachi's HM51258P-1X:

Hitachi is apparently the first chip vendor to extend the page mode concept to its next plateau: the use of static mode CAS access. This means that after #RAS is asserted, and then #CAS, that as long as #RAS and #CAS are both low, then the #CAS address lines—all nine of them—can access the DRAM as though it were a 512-bit static RAM. That means that CAS does not have to go high, then low again, for each bit read. And that saves lots of time. The –12 part, for instance, features the usual 120nsec access time from #RAS, but subsequent fetches within the same 512-bit "page" can be performed in as little as 55nsec! This represents, while staying within that one 512-bit page, more than a 2–1 speed increase!

Others are believed ready to follow Hitachi into the static CAS DRAM business. And we have a new way to provide the insatiable need for high-speed DRAM for tomorrow's microprocessors.

ACORN's Risc Micro:

The *Electronic New* story provided some interesting details but contained at least one obvious error, so be warned. (The article asserted that both the VAX 11/750 and the Nat Semi 32016 had a 4Mbyte/sec bus bandwidth. The VAX we dunno about, but the 10MHz 32016 requires 4 clocks per 16-bit bus cycle, and thus has a 5Mbyte/sec bus bandwidth.

The ACORN RISC machine is now based on CMOS with 3 micron design rules. Naturally, they plan to shrink the chip to 2 microns the day after tomorrow. The chip apparently has 25 each 32-bit registers, about 50% more than the 68000. Our guess is that this chip has a bit less oomph than the Fairchild RISC machine.

What fascinates us about the ACORN is this innocuous sentence in that article: "The [ACORN RISC] has an optional page mode that can increase data-transfer rates another 30%." (That's how ACORN came up with that 18Mbytes/sec bus bandwidth, in case you were wondering.)

It also means that the ACORN RISC Machine (the "ARM"—shudder) is the first microcomputer to include on-chip logic to take advantage of the page mode of ordinary DRAMs. Imagine when advanced versions of the 68020, 386, possibly even the now-mythological Z80000, become available with built-in logic to take advantage of static column DRAMs and start running at 25MHz with no wait states! With a 32-bit bus! Yum!

[Until that happens we will all have to use lots of PALs (Programmable Array Logic, a trademark of MMI) to implement the page-mode logic for our high-speed micros—unless we use the ACORN (ugh!) ARM.]

The Paging Game

This game's most recent appearance was in an Australian publication called *CSIRNET NEWS*, #182 (!), April 1985. The subtitle is "Any similarities between this game and the workings of a certain CSIRNET host computer are purely coincidental—aren't they?" We'd like to give credit, but the author is not listed.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

COMBINE THE
RAW POWER OF FORTH
WITH THE CONVENIENCE
OF CONVENTIONAL LANGUAGES

HS/FORTH

Why HS/FORTH? Not for speed alone, although it is twice as fast as other full memory Forths, with near assembly language performance when optimized. Not even because it gives MANY more functions per byte than any other Forth. Not because you can run all DOS commands plus COM and EXE programs from within HS/FORTH. Not because you can single step, trace, decompile & dissassemble. Not for the complete syntax checking 8086/8087/80186 assembler & optimizer. Nor for the fast 9 digit software floating point or lightning 18 digit 8087 math pack. Not for the half megabyte LINEAR address space for quick access arrays. Not for complete music, sound effects & graphics support. Nor the efficient string functions. Not for unrivaled disk flexibility — including traditional Forth screens (sectored or in files) or free format files, all with full screen editors. Not even because I/O is as easy, but far more powerful, than even Basic. Just redirect the character input and/ or output stream anywhere — display, keyboard, printer or com port, file, or even a memory buffer. You could even transfer control of your entire computer to a terminal thousands of miles away with a simple >COM <COM pair. Even though a few of these reasons might be sufficient, the real reason is that we don't avoid the objections to Forth — WE ELIMINATE THEM!

Public domain products may be cheap; but your time isn't. Don't shortchange yourself. Use the best. Use it now!

HS/FORTH, complete system: $395. with "FORTH: A Text & Reference" by Kelly and Spies, Prentice-Hall and "The HS/FORTH Supplement" by Kelly and Callahan

Visa Mastercard

HARVARD SOFTWORKS
PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

Circle 42 on the reader service card

1. Each player gets up to 8,388,608 Things in the version 4 game. In older games, each player got up to 262,144 Things.
2. Things are kept in Crates that hold 512 Things each. Things in the same Crate are called Crate-mates.
3. Crates are stored either in the Workshop or a Warehouse. The Workshop is almost always too small to hold all the Crates.
4. There is only one Workshop but there may be several Warehouses. Everybody shares them.
5. Each Thing has its own Thing number.
6. What you do with a Thing is to zark it. Everybody takes turns zarking.
7. You can zark only your Things, not anybody else's.
8. Things can be zarked only when they are in the Workshop.
9. Only the Thing King knows whether a Thing is in the Workshop or a Warehouse.
10. The longer a Thing goes without being zarked the grubbier it is said to become.
11. The way you get Things is to ask the Thing King. He gives out Things only in full Crates at a time.
12. The way you zark a Thing is to give its Thing Number. If you give the number of a Thing which happens to be in the Workshop, it gets zarked right away. If it is in a Warehouse, the Thing King packs the Crate containing your Thing back into the Workshop. If there is no room in the Workshop, he first finds the grubbiest Crate in the Workshop, whether it is yours or somebody else's, and packs it off with all its Crate-mates to a Warehouse. In its place he puts the Crate containing your Thing. Your Thing then gets zarked and you never knew that it wasn't in the Workshop all along.
13. Each player's Things have the same numbers as everybody else's. The Thing King always knows who owns what Thing and whose turn it is, so you can't ever accidentally zark somebody else's Thing even if it has the same Thing Number as one of yours.
14. If the Thing King must move a Crate from the Warehouse to the Workshop, to let you zark it, he first sends you to sleep so you won't mind the wait.

Notes:

1. Traditionally, the Thing King sits at a large, segmented table and is attended to by pages (the so-called 'Table pages') whose job it is to help the Thing King remember where all the Things are, who they belong to, and how grubby they are.
2. One consequence of Rule 13 is that everybody's Thing Numbers will be similar from game to game, regardless of the number of players.
3. The Thing King has a few Things of his own, some of which move back and forth between Workshop and Warehouse just like anybody else's. Some, however, are just too heavy to move out of the Workshop.
4. With the given set of rules, oft-zarked Things get kept mostly in the Workshop while little-zarked Things get kept mostly in a Warehouse. This is efficient stock control.
5. Sometimes even the Warehouses get full. The Thing King then has to start piling Crates on the dump out back. This makes the game slower because it takes a long time to get Crates off the dump when they are needed in the Workshop. The Thing King tries to select the grubbiest Crates in the Warehouse to send to the dump in his spare time, but sometimes even this doesn't help, and the Thing King has to take more time away from players to move Crates. This is caused by a player trying to zark so many Things at once that even Crates which aren't grubby at all have to be moved into the Warehouse. This is called thrashing, and at times like these the game

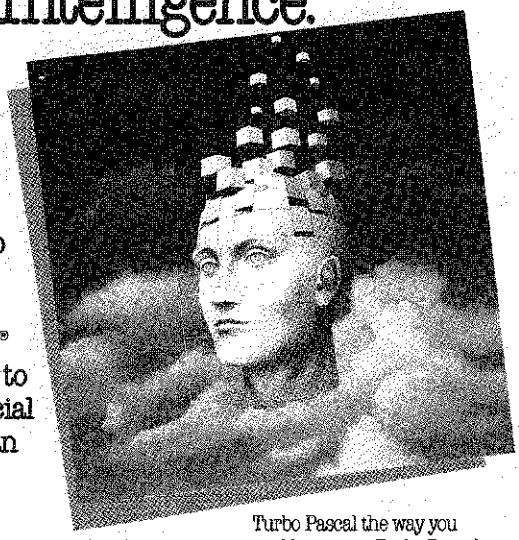
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Step-by-step tutorial, demo programs with source code included!

Borland introduces Turbo Prolog, the natural language of Artificial Intelligence.

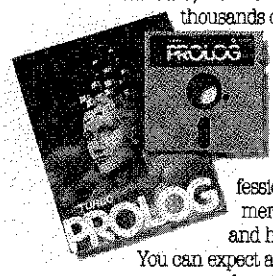
Prolog is probably the most powerful computer programming language ever conceived, which is why we've made it our *second* language—and "turbocharged" it to create Turbo Prolog.™

Our new Turbo Prolog brings supercomputer power to your IBM® PC and introduces you step-by-step to the fascinating new world of **A**rtificial **I**ntelligence. And does all this for an astounding $99.95.



Turbo Prolog is to Prolog what Turbo Pascal® is to Pascal!

Our Turbo Pascal astonished everyone who thought of Pascal as "just another language." We changed all that—and now Turbo Pascal is the de facto worldwide standard, with hundreds of thousands of enthusiasts and users in universities, research centers, schools, and with professional programmers, students, and hobbyists.



You can expect at least the same impact from Turbo Prolog, because while Turbo Prolog is the most revolutionary and natural programming language, it is also a complete development environment—just like Turbo Pascal.

Turbo Prolog radically alters and dramatically improves the brave new world of artificial intelligence—and invites you into that fascinating universe for a humanly intelligent $99.95.

Even if you've never programmed before, our free tutorial will get you started right away

You'll get started right away because we have included a complete step-by-step tutorial as part of the 200-page Turbo Prolog Reference Manual. Our tutorial will take you by the hand and teach you everything you're likely to need to know about Turbo Prolog and artificial intelligence.

For example: once you've completed the tutorial, you'll be able to design your own expert systems utilizing Turbo Prolog's powerful problem-solving capabilities.

Think of Turbo Prolog as a high-speed electronic detective. First you feed it information and teach it rules. Then Turbo Prolog "thinks" the problem through and comes up with all the reasonable answers—almost instantly.

If you think that this is amazing, you just need to remember that Turbo Prolog is a 5th-generation language—and the kind of language that 21st century computers will use routinely. In fact, you can compare Turbo Prolog to Turbo Pascal the way you could compare Turbo Pascal to machine language.

You get the complete Turbo Prolog programming system for only $99.95

You get a complete Turbo Prolog development system including:

- The lightning-fast Turbo Prolog incremental compiler and the interactive Turbo Prolog editor.
- The 200-page reference manual which includes the step-by-step Turbo Prolog tutorial.
- The *free* GeoBase™ natural query language database including commented source code on disk—ready to compile. GeoBase is a complete database designed and developed around U.S. geography. It includes cities, mountains, rivers, and highways, and comes complete with natural query language. Use GeoBase immediately "as is," or modify it to fit your own interests.

So don't delay—don't waste a second—get Turbo Prolog now. $99.95 is an amazingly small price to pay to become an immediate authority—an instant expert on artificial intelligence! The 21st century is only one phone call away.

Turbo Prolog 1.0 Technical Specifications

Programming System Features

- ☑ *Compiler:* Incremental compiler generating native in-line code and linkable object modules. The linking format is compatible with the PC-DOS linker. Large memory model support. Compiles over 2500 lines per minute on a standard IBM PC.
- ☑ *Interactive Editor:* The system includes a powerful interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code. At run-time, Turbo Prolog programs can call the editor, and view the running program's source code.
- ☑ *Type System:* A flexible object-oriented type system is supported.
- ☑ *Windowing Support:* The system supports both graphic and text windows.
- ☑ *Input/Output:* Full I/O facilities, including formatted I/O, streams, and random access files.
- ☑ *Numeric Ranges:* Integers: −32767 to 32767; Reals: 1E−307 to 1E+306
- ☑ *Debugging:* Complete built-in trace debugging capabilities allowing single stepping of programs.

YES! I want the best

Turbo Prolog at only:

$99.95

To order by phone, or for a dealer nearest you,

Call (800) 255-8008

in CA call (800) 742-1133.

Send me ___ Turbo Prolog at $ _____

Outside USA add $10 per copy
CA and MA res. add applicable sales tax $ _____

Amount enclosed: $ _____

This price includes shipping to all US cities

Payment: VISA MC Bank Draft Check

Credit card expiration date: ___/___

Card # |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

*You must have an IBM or true compatible running DOS 2.0 or later.***
My computer's name and model is:

_____

The disk size I use is: ☐ 3½" ☐ 5¼"

NOT COPY PROTECTED
****60-DAY MONEY-BACK GUARANTEE***

Name: _____

Shipping Address: _____

_____

City: _____

State: _____ *Zip:* _____

Telephone: _____

CODs and purchase orders WILL NOT be accepted by Borland. Outside USA make payment by credit card or International Postal Money Order.

**YES, if within 60 days of purchase this product does not perform in accordance with our claims, please call our customer service department and we will gladly arrange a refund.*

***** Minimum system requirements:***
IBM PC, XT, AT, PCjr, and true compatibles
384K RAM

SU36

BORLAND INTERNATIONAL
Vive la différence

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373

Other Borland Products include Turbo Pascal; Turbo Tutor; Turbo Lightning; Turbo Database Toolbox; Turbo Graphix Toolbox; Turbo Editor Toolbox; Turbo GameWorks; SuperKey; SideKick; SideKick, The Macintosh Office Manager; Reflex, The Analyst; and Traveling SideKick—all of which are registered trademarks or trademarks of Borland International, Inc. or Borland/Analytica, Inc.
Turbo Prolog and GeoBase are trademarks and Turbo Pascal is a registered trademark of Borland International Inc. IBM and AT are registered trademarks of International Business Machines Corp. Copyright 1986 Borland International BI-1046D

Circle 3 on the reader service card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

is more for a fool than the Thing King.

6. From note 5, a player can figure out that if he has a lot of Things to zark, it is best to zark only a small number of Things at once, and to get a Thing's zarking done with quickly. That way, the Thing King doesn't have to do as much thrashing and can reward the player by putting him to sleep less often. Players who do this win the game, and help other players win too.

Our Software Viewpoint:

We are very commonly accused of suggesting that everything be written in assembly. That is not even remotely true. Here is what we DO believe:

ANY SOFTWARE PACKAGE AIMED AT THE MASS PERSONAL COMPUTER MARKETPLACE WHICH IS WRITTEN IN A HIGH LEVEL LANGUAGE WILL BE REJECTED BY THE MASS COMPUTER MARKETPLACE IF AN EQUIVALENT SOFTWARE PACKAGE IS AVAILABLE WHICH IS WRITTEN IN ASSEMBLY.

There are MANY examples to prove the above, and NO examples contrary to the above. Despite the obvious nature of the above rule, there are still companies which are coming to the mass-market with operating systems written in C (such as DRI and any company flogging UNIX) to compete against MS-DOS. The UNIX folk have the excuse that they are not familiar with the mass personal computer marketplace but DRI has NO excuse. We have to believe that DRI has a corporate death-wish, one which seems to be nearing fulfillment.

On the other hand, Microsoft's MacBASIC, which is written in a high-level language and hence slows the Mac's 68000 down to the stately pace of an Apple II, is (modestly) successful only because there is no competing product written in assembly.

A similarly slow product, slow due to being written in a high level language (Pascal), was CONTEXT'S MBA integrated spreadsheet. You know what happened to MBA!

Programmer Cattle-Pens:

We are well aware that there are certain software projects, such as those requiring 250 programmers and 36 months, which absolutely require a high level language such as Ada or COBOL. However, we are not personally involved in such a project, nor are we interested in such projects, and we don't think that any such projects are relevant to the person who wants to use a computer exclusively for personal reasons. We implicitly ignore such cattle-pen programming projects.

Our Software Agenda:

This is going to surprise you. Ease of use, meaning interactivity, is at the top of our software agenda, just barely ahead of speed. Or haven't you noticed all the work we have done on the interactive high level language HALGOL?

If speed were at the top of our software agenda, we could go out there someplace and buy a FORTRAN compiler. The problem is, we HATE compilers! Which means that we are

Atron's PC/AT Bugbusters

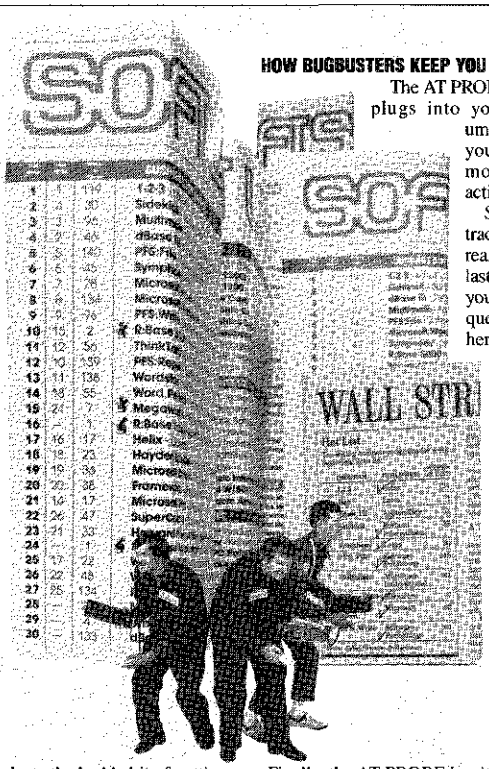
Hardware-assisted Software Debuggers for Bullet-proof PC/AT-based Products

A BUGBUSTER STORY

Brad Crain, a project manager at Software Publishing (the people who developed both PFS:WRITE and PFS:FILE), relates the following: "On Friday, March 22, 1985, I was about to get on an airplane with Jeff Tucker, who was co-author of PFS:WRITE with me, and fly to IBM's Boca Raton, Florida facility. For a week, we had been unsuccessfully trying to isolate a bug in a new software product. In a last, desperation move, I set up an early-Saturday morning appointment with ATRON.

"Three of us walked through ATRON's door at 8:00 the next morning. Using ATRON's hardware-assisted debugging tools, we had the problem identified and fixed by 10:30AM."

Mr. Crain concludes: "We'd never have found the bug with mere software debuggers, which have the bad habit of getting over-written by the very bugs they're trying to find. It doesn't surprise me that almost all the top-selling software packages were written by ATRON customers. Now that they've broadened their PC family of debuggers to include a PC/AT debugging tool, those of us seriously into 80286 development are greatly relieved."



HOW BUGBUSTERS KEEP YOU FROM GETTING SLIMED

The AT PROBE is a circuit board that plugs into your PC/AT. It has an umbilical which plugs into your 80287 socket and monitors all processor activity.

Since AT PROBE can trace program execution in real time, and display the last 2048 memory cycles, you can easily answer the questions: "How did I get here?" and "What *are* the interrupts doing?"

It can solve spooky debugging problems. Like finding where your program overwrites memory or I/O - impossible with software debuggers.

You can even do source-level debugging in your favorite language, like C, Pascal or assembler. And after your application is debugged, the AT PROBE's performance-measurement software can isolate your application's bottlenecks.

Finally, the AT PROBE has its own 1-MByte of memory. Hidden and write-protected. How else could you develop that really large program, where the symbol table would otherwise demand most of your PC/AT memory.

ARE YOU TRYING TO DO SOMETHING SCAREY?

Like developing your AT-based software product in the dark? Without professional debugging tools?

Seven of the ten top-selling software packages listed by the *THE WALL STREET JOURNAL** were produced by ATRON customers. The PC PROBE™ bugbuster ($1595) accounts for much of this success. Now that the PC/AT is the new standard for advanced commercial and scientific development, ATRON is proud to announce the AT PROBE™ bugbuster ($2495). It has even more debugging capabilities than the PC Probe.

BORLAND'S PHILIPPE KAHN: "THERE WOULDN'T BE A SIDEKICK™ WITHOUT ATRON'S DEBUGGERS."

So why waste more time reading though your program listing for the ten thousandth time, trying to find why your program starts howling with every full moon. Be like BORLAND, get your Atron bugbuster today and bust bugs tomorrow.

atron
THE DEBUGGER COMPANY
20665 Fourth Street • Saratoga, CA 95070 408/741-5900

*WSJ, June 24, 1985, reporting Softsel figures. © 1985 by ATRON. PC PROBE™ and AT PROBE™ ATRON. SIDEKICK™ Borland. IBM Corp. owns numerous trademarks. Ad by TRBA.

Circle Number 1 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

never going to personally cozy up to FORTRAN, or MODULA, or even C. On the other hand, this is clearly a question of personal preference. We have no objection whatever if YOU want to program in a compiled language. Each cat its own rat.

But we like developing programs in an interactive environment, and that means BASIC to most of us. (Exceptions include APL and those folks who were lucky enough to buy a SUPERPET while they were still in production; the SUPERPET included interactive versions of all the popular computer languages including COBOL!) Unfortunately, all of the available interactive BASICs written for the 68000 are written in a high level language. This left us with zero choices.

ZERO choices? Really? Hmmm . . .

Well, we COULD write our own fast 68000 BASIC . . .

And we have. HALGOL is now sufficiently complete that we are using it to perform routine computing tasks of the sort which we used to perform on our 1972-vintage Wang 2200. In the past couple of months, we have spent more time programming in HALGOL than in assembly. And because interactivity is only slightly ahead of speed on our software agenda, HALGOL runs about as fast as is possible using the 68000 (instead of, for instance, a CRAY-2). In fact, we made the claim a few issues back that HALGOL is the fastest microprocessor-based interactive BASIC-like language in the world, and not one of you readers, including you full-time computer professionals, has demurred.

Is this because we are a great programmer? No, it is just that nobody else has caught on to the fact that the 68000 world needs a BASIC which wins in assembly instead of loses in HLL. If everybody else continues to keep that HLL mentality, and if somebody will make a 68000-based personal computer which is not impossibly difficult to program, then we may profit from HALGOL.

The fact that speed is very close to interactivity on our personal agenda has some interesting consequences. It caused us to write our interactive language as an incremental compiler, not as an interpreter. The good news is that the result is a language which runs 50 to 150 times faster than Applesoft (or comparable 8080-based BASICs) instead of the 20 times faster which could be expected of a no-wait-state 12.5MHz 68000. The bad news is that HALGOL syntax is not quite as flexible as that provided by leisurely interactive BASICs (or BASICs which compile from source text corresponding to leisurely interactive BASICs).

What all this proves is that THERE IS NO SUCH THING AS ZERO CHOICES OF COMPUTER LANGUAGES; you can always write your own. WE DID! Of course, it is helpful if you have a fairly long attention span . . .

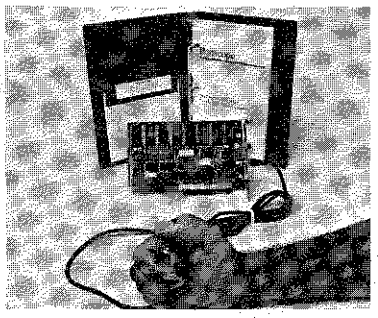
Incremental Compilers:

Incremental compilers are slower than compilers and faster than interpreters. *Electronics Week* featured incremental compiling on its front page as a hot new technique. *DATAMATION*, in Jan '70, carried a TUTORIAL on incremental compiling (pp 129-135)! Tutorials, as you all know, are written by authorities on mature technologies. So incremental compiling was a mature technology in 1970! All things considered, it would be a good idea that we not claim to have invented incremental compiling, so we won't.

(We also did not invent threaded code, but neither did Moore and Rather. The first commercial use of threaded code was on a DEC FORTRAN compiler, back in the '60s. HALGOL is an incremental compiler which compiles threaded code. The reason HALGOL uses threaded code is simply that the 68000 executes threaded code faster than an equivalent series of subroutine calls.)

Let us explain, in a very simple fashion, why an incremental compiler is much faster than an inter-

Periscope Delivers Professional Debugging Power



Models I and II are the same except Model I includes a write-protected RAM board. If you buy Model II, then find you need the board's protection, you can trade up to Model I. Both models include the break-out switch.

GET YOUR PROGRAMS WORKING FAST! "A problem we had for three weeks was solved in three hours," says Wade Clark of MPPi, Ltd. The break-out switch lets you stop your system anytime, even when it's hung, to see what's going on. Symbols, plus source debug for high-level languages, save you hours. Powerful breakpoints on registers, memory, I/O ports, and code help you find bugs fast.

MAKE YOUR SOFTWARE RELIABLE! "I can't live without it!! BRIEF, a text editor my company wrote, would not be as stable as it is today without Periscope," writes David Nanian. Periscope helps you find ALL your bugs so your software is reliable. The latest release includes traceback, which shows up to 2,016 previous instructions, EGA support, user exits, text file viewing, 8087/80287 status display, and much more.

DEBUG PROGRAMS OTHER DEBUGGERS CAN'T! Debug memory-resident programs, keyboard routines, interrupt-driven programs, device drivers, and DOS. Periscope gives you the best crash recovery capability available. "Periscope is rock solid," says user Doug Roberts.

GET THE BEST VALUE! Time and again users tell us Periscope pays for itself in a few hours after they begin using it. This means no professional software developer can afford NOT to try it!

PERISCOPE I: Board, Switch, Software, Manual, Reference Card ..... ONLY $295

PERISCOPE II: Switch, Software, Manual, Reference Card ..... JUST $145

PERISCOPE REQUIRES: An IBM PC, XT, AT or close compatible; DOS 2.0 or later; 128K RAM; one Disk Drive; and an 80-column Monitor.

Call Now, Toll-Free, For More Information Or To Place Your Order

MasterCard **800-722-7006** VISA

30-Day Money-Back Guarantee

Get Your Programs Up and Running;

UP PERISCOPE!

Data Base Decisions • 404 256-3860
14 Bonnie Lane • Atlanta, GA 30328

Circle Number **6** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

preter but slower than a compiler. Take expression evaluation, for instance:

```
18 + B - C
```

An interpreter has to examine that very simple expression byte by byte to determine which parameters are constants, which are variables, and which are mathematical operators, and must test to be certain that mathematical precedence is followed correctly. In the above expression, there are no precedence problems, but the interpreter cannot know that, so it has to test anyway. Having identified "18" as a constant, it has to turn that constant into a binary floating point value. Having identified which parms are variables, it must perform a linear search through a variable table to locate the address of the variable value. And an interpreter does not learn; the next time it encounters that same expression it has to perform all those tasks all over again.

A compiler performs many of the above tasks in advance, at compile time. A compiled program containing the above expression will already have evaluated mathematical precedence and rearranged the expression, if necessary, for optimum evaluation without unnecessary loads and stores into temporary locations. The constant "18" will already have been converted into a binary floating point number. And the compiler will already have the location (address) of the variables on hand. Hence, many of the burdensome tasks performed at run time by an interpreter are eliminated, and the compiler runs much faster than an interpreter.

An incremental compiler operates the same way as a compiler, except that it compiles a variable table offset instead of the actual address of the variable, a constant table offset of the F.P. constant, a label address table offset instead of the actual address of the label, etc. At run time, the address must be calculated by adding the base address of the table in question to the offset. Therefore, an incremental compiler is slower than a true compiler.

Programmer/Talent:

(Earlier today we found ourself and a longtime UNIX guru both calling programmers "talent," but from differing perspectives. The following is a rough rendition of the conversation which ensued, with neither party being further identified for reasons which will become obvious.)

"The existing method of employing programmers is all wrong. Businesses like to pretend that all employees work 8 hours a day, 5 days a week, 50 weeks a year. Programmers don't do that. They CAN'T! Good programmers work sporadically or episodically, in long, intense spurts. And then they sit back and recoup their creative juices for a while before the next episode. The point is, a programmer is not a ditch digger or a bank teller. Diggers and tellers can always force themselves to get up and go to work, even on days they feel blah or grundgy. On those days when a programmer feels blah or grundgy, no useful programming is going to get done. In fact, if a forced attempt is made to program anyhow, some active damage may well be done to their project which will require correcting in the future."

"The American business system, based on the 8-hour 5-day week, does not make allowances for those days when "talent" is not going to be productive, nor does it support the episodic work binges which talented programmers go on and which do not conveniently fit into the time slot of 8 am to 5 pm. This may have something to do with widely reported poor programmer productivity. Round pegs do not fit square holes!"

"I know something else which might be relevant to that widely reported poor programmer productivity. Not only are their work habits not compatible with business hours, but it may not even be possible to actually hire a programmer in the usual

Add Data Security to Your C Programs

The SECURITY LIBRARY™

Add Fast or Thorough Encryption or Compression to Your Programs WITHOUT Royalties.

Building security into your programs is easy with The Security Library from Solution Systems. The package includes a subroutine library with algorithms for data encryption and compression, valuable related utilities, a comprehensive discussion of security methods, and demo programs.

The Security Library allows you to:

- Keep Files Secure on a Multi-User Network
- Speed up Data Transmission and Communications Through Data Compression
- Control Access Based on Privileges given to a Hard Disk or LAN User

Algorithms provided for encryption include The National Bureau of Standards' Data Encryption Standard (DES) and the Vernam and Vigenere ciphers. You can encrypt a 10K file in 3 seconds with one method or 50 seconds with another. Compression routines typically decrease file sizes and transmission times by 25% or more.

Using the C source code, you can build the safest and most popular methods of protecting your data directly into your programs, you pay NO royalties.

Order the Security Library for Microsoft C and Computer Innovations' C86. Works on MS-DOS 2.0.

Only $125 for Object
$250 for source & object

Full refund if not satisfied during first 30 days

Solution Systems™

335 Washington St.
Norwell, MA 02062
617-659-1571
800-821-2492

Circle 47 on the reader service card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

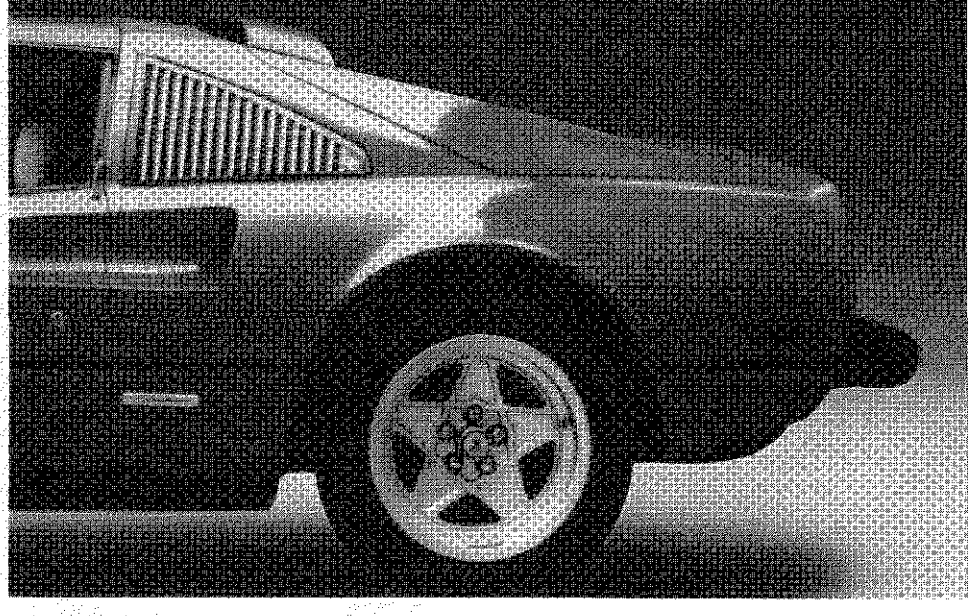
sense. It seems that all programmers really believe that they are going to get rich by developing a software package which THEY, and not their employers, own."

"What this means is that the programmers go home, have supper, and then program on their own packages until 1 am (or even later for younger programmers). Since programmers are talent and not bank tellers, they show up on the job next day in body but not in spirit. Talent has just so much creative juices, and most of these creative juices are expended at home on the software project which is going to make them rich. Some programmers become even bolder and more contemptuous of their employers and spend long telephone conversations during the day supporting their evening programming projects. The first time this happened to me was in 1973. These yahoos seem to think that an employer can't or won't notice what is going on."

"Some programmers will haul down $50,000 from their employer and still go home at night and piss away their creative juices on their very own proprietary software package. And the industry wonders why programmer productivity is so low? When you consider how many programmers there are—a million? millions?—and how very few have succeeded with these home projects, you would think they would concentrate on convincing their employers to give them $10,000 raises. But no, that's the way a bank teller would think. Talent is always thinking about grabbing the big brass ring because *that is how talent thinks*!"

"In the meantime, there is precious little creative juice left for the employer, and so the industry average is ten lines of debugged, documented code per programmer per 8-hour day."

"So programmers are hired the wrong way. Nobody hires a portrait-painter by the hour; you hire him or her to paint a portrait of a given size for a fixed fee. Programmers should be hired for a fee to program a given task, with a substantial portion of the



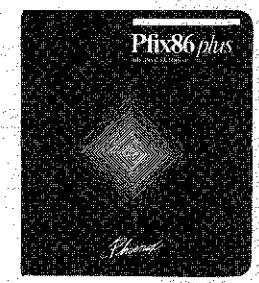
New Advanced Symbolic Debugger Handles Complex Overlayed Programs.

Phoenix's new Pfix™86 Plus for MS-DOS®/PC DOS environments features configurable menus and keys, tracebacks, EGA support, and the ability to stop, save, and restore a debugging session at any time.

You can debug without a listing, since you can see and enter symbolic names or absolute addresses in breakpoints, data displays, expressions, or with the in-line assembler.

Display source files, disassembled object, data area, stack, breakpoint settings, CPU and coprocessor registers, simultaneously. Set breakpoints in the source file window. Synchronize the source display with the disassembled object. Add symbols incrementally during the debug session. And write the disassembly to disk.

Pfix86 Plus supports 80xx and 80xxx CPUs and floating point processors, EGA boards, user-assignable string and numeric variables, up to 100-step tracebacks, and logs to disk or printer.



$395 complete.

Send for your Pfix86 Plus information kit today. Call or write:
Phoenix Computer Products Corporation
320 Norwood Park South
Norwood, MA 02062
(800) 344-7200. In Massachusetts (617) 762-5030

Programmers' Pfantasies™
by

Phoenix

Programmers' Pfantasies is a trade mark of Phoenix Computer Products Corporation.
Pfix86 Plus is a trade mark of Phoenix Software Associates, Limited.
MS-DOS is a registered trademark of Microsoft Corporation.

Circle Number 18 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

fee withheld until the task is demonstrably completed. They most certainly should not be required to confine their efforts to the non-talent 8 am to 5 pm workday. For some days in a row, the programmers may not feel the creative juices flowing. Fine; he or she doesn't get paid. Some days the juices flow freely, and a programmer may complete a task very, very quickly in terms of what a manager of bank tellers might think of as hours per dollar. Great! Pay the agreed fee and be glad the work was completed so soon!"

"I think you are overstating your case. Programmers are not chattel slaves, and their employers are not slavemasters. There is ample legal precedent that an employee's time is his or her own outside of working hours. Moonlighting is a way of life for lots of folk and a sometimes way of life for others. I agree that programmers in general do additional work in the evening, but I think it is just the sort of thing you are talking about: contract work for a fixed fee. And yes, it does sop up the creative juices."

"I think you will have a tough job on your hands trying to keep programmers around full time if you pay them on a job basis. They'd rather hire on with your competitor for a nice salary so they can support themselves while they work on something else in the evening."

"In that case, I'll hire my competitor's programmers on a job basis to do my programming in the evening!"

We bet none of you readers thought that we and a UNIX guru would ever agree on anything.

Aloha, 32032:

Gould is the latest company to dump a major development project based on the 32032. Texas Instruments, which is the second source for the 32032, has quietly folded the 32000 division into its programmable controller products division. Inmos, in promoting its Transputer, asserts that the Transputer is four times as fast as the 68020, nine times faster than the 80286, and twelve times faster than the 32032. Now, I don't believe that Transputer comparison for a minute (as a GENERAL PURPOSE computer), but Inmos is a disinterested observer of 68020, 80286, and 32032 performance.

What Inmos annnounced (*Electronic News* 7 Oct p. 20) is that the 68020 is 2.25 times faster than an 80286 and 3.0 times faster than the 32032! And the 68020 is available NOW, and almost entirely without bugs. The 32032 is still heavily bug-ridden, as in multiple-page bug lists. Can it be surprising that 32032 projects are being cancelled right and left? Oh, yes!

Col Rada (that's an East Indian name), who was in charge of 32000 series marketing and promotion, just uh, QUIT—without giving notice! QUIT? Hmmm . . . .□

King Computer Vocabulary #2 Entry

SUPER-CHARGE your directories for only $39.95

Say goodbye to path names and unleash the full power of DOS on your IBM Personal Computer. With **SuperPATH™** from Martin Scot, you can run any program from any directory, from any disk, anytime.

SEE THE BIG PICTURE.
Tired of playing hide and seek with your files? **SuperPATH** tells you where your files are, instantly. Anywhere on your disk.

EXPLOSIVE PERFORMANCE.
Program your directories. To display just the files you want to see, in any format, sorted or double sorted. To set up your keyboard macros for each new directory you enter. To back up your files automatically when you leave a directory.
Or, you can use **SuperPATH**'s batch building option to create your own file management utilities.

NO RISK.
Comes with a 30 day money-back guarantee.

CALL (206) 527-9605 to order SuperPATH.

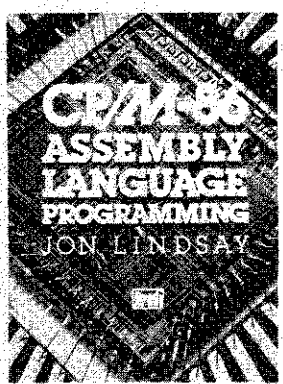
MARTIN SCOT
Development Corporation
4515 Purdue N.E. Seattle, WA 98105

Circle Number **27** on the Reader Service Card

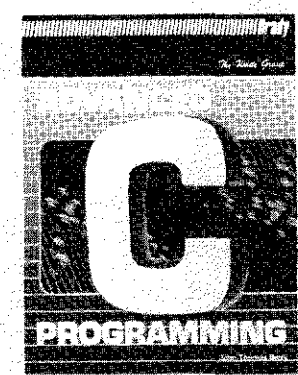
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BRADY Speaks Your Language

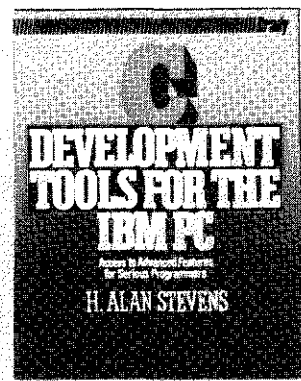
We can help you buff up your BASIC...Put a finish on your FORTH... Conquer C...Succeed with Assembler...And more! Just call toll-free or use the coupon to order today!



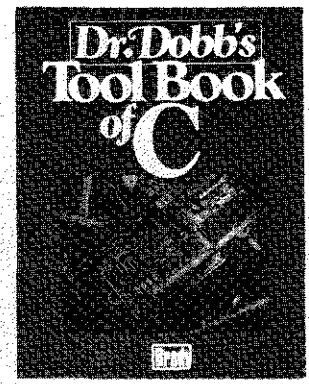
1. The perfect intro to CP/M-86 even if you've never programmed in assembly language before. Includes sections on sequential file handling and memory manipulation, and more. $21.95



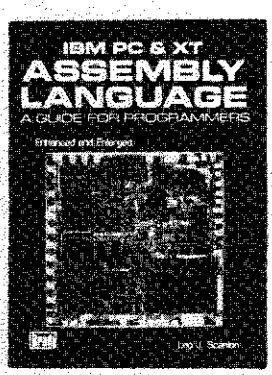
2. Beyond the basics, this guide explores new structured concepts to analyze and solve problems in C with tools of modularity, input and output functions, arrays and structures, and bit manipulation. $21.95



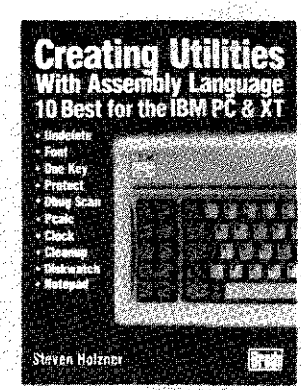
3. The special focus is on screen management and data retrieval. You'll find a library of reusable C software tools to help create interactive systems, including file management, screen forms, etc. $21.95



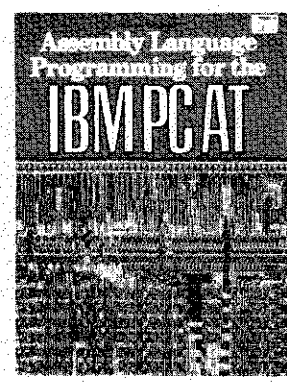
4. The best C articles from the highly respected *Dr. Dobb's Journal* dealing exclusively with C language and programming techniques. $24.95



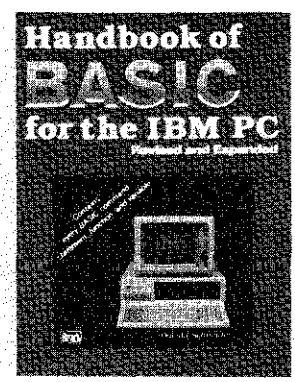
5. The author of our best-selling assembler books now demonstrates his detailed and accurate style on the 80286 chip. $21.95



6. For the more advanced user familiar with Assembly language, here's an opportunity to unleash the power of 10 DOS-enhancing programs. $21.95



7. Our best-selling assembler book has been made even better! It now includes 30 assembler Macros and version 2.0 of the IBM Assembler. $21.95



8. *PC* magazine calls it: "A truly remarkable book...A treasure trove of useful programming information on the IBM PC." $22.95

Now at your book or computer store.
Or order toll-free today: **800-624-0023**
In New Jersey: 800-624-0024

BRADY COMMUNICATIONS COMPANY, INC.
c/o Prentice Hall
P.O. Box 512, W. Nyack, NY 10994

Circle the numbers of the titles you want below. (Payment must be enclosed; or, use your charge card.) Add $1.50 for postage and handling. Enclosed is check for $_____ or charge to
☐ MasterCard ☐ VISA

Acc't # _____ Exp. date _____
Signature _____
Name _____
Address _____
City _____ State _____ Zip _____
(New Jersey residents, please add applicable sales tax.)
Dept. 3
GR-BLPJ-BF(8)

1 (0-89303-390-1) 2 (0-89303-473-8) 3 (0-89303-612-9) 4 (0-89303-599-8)
5 (0-89303-575-0) 6 (0-89303-584-X) 7 (0-89303-484-3) 8 (0-89303-510-6)

Brady

Circle 37 on the reader service card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SUN MICROSYSTEMS' NEXT GENERATION PC

by Frank D. Greco

When Is an Engineering Workstation not an Engineering Workstation?

For the uninitiated, Sun Microsystems, Inc., manufactures high resolution, large screen, monochrome or color bit-mapped, UNIX-based and easily networked workstations. Many of you PCers out there might say "Who cares?" Well, if you haven't noticed, the workstation market is looming over us. Have you noticed that many IBM PC ATs are now used for CAD (Computer Aided Design), CAM (Computer Aided Manufacturing), window-driven multi-tasking, document publication, and most importantly, Artificial Intelligence? Every company is noticing this. Do you think it is coincidence that Apple Computer, a microcomputer vendor, announces a laser printer along with a bit-mapped Mac? Do you think that DEC arbitrarily released the high resolution bit-mapped VAXStation (based on the MicroVAX II) soon after the MicroVAX itself was first demonstrated? Do you think IBM announced their PC RT, a RISC (Reduced Instruction Set Computer) based bit-mapped workstation, just for fun? Do you think that all these IBM PC AT graphics board vendors are busy at work just to keep off the streets? Engineers are not the only ones who want to improve their productivity; bit-mapped workstations (notice that I did NOT say *engineering* workstations) will find their way into the office within the next few years. And Sun machines will find their way into many of these offices.

Everything Under the Sun? or WYSIWYG? or How about IBIWISI (I'll Believe It When I See It)?

Since AT&T has been blasted for encouraging many different varieties of UNIX, they sought the help of Sun Microsystems in combining the two most popular versions of UNIX: System V and Berkeley's BSD 4.2. Since Sun Vice President Bill Joy was the primary architect of Berkeley UNIX (and the author of the 'vi' screen editor and the 'csh' shell) and Sun is one of the fastest growing startups in the Valley's history, it's fairly obvious why AT&T went to Sun. Each company will come out with a version of UNIX that incorporates features from both versions. The phone company will produce a version of System V with 4.2's networking (which probably includes Sun's Network File System), real-time functionality (IBIWISI), and graphics (IBIWISI because Sun uses a proprietary graphics/window system). Sun's part of the agreement will include selling a version of the joint operating system that will allow application programmers to write for System V and be assured that their programs will run on a Sun 4.2-based workstation.

Interprocess Communication Schemes Compatible? IBIWISI

I wonder how Sun will get AT&T's System V-based interprocess communication schemes to work under BSD 4.2. It does not appear to be easy. For example, Berkeley relies on entities called "sockets" to be the main communication connections between processes. Pipes, which connect the standard output of one process with the standard input of another process, are implemented using sockets in the Berkeley implementation of UNIX. Will AT&T use sockets to implement pipes in System V? That's chancy because AT&T is betting heavily on Dennis Ritchie's "Streams" (see "Redirections," *PJ*, Vol. 3, No. 6), which may be more elegant, reliable, and potentially easier to program than sockets. The kludgey solution is to incorporate every IPC scheme into UNIX. However, this answer implies that the kernel (or nucleus) of the UNIX operating system becomes physically larger and increasingly non-elegant. So, will my programs that use Berkeley-based sockets run on System V-based UNIX? My guess is that AT&T will incorporate sockets, but the implementation will be a kludge; it's still a jump ball.

Timing Compatible? IBIWISI

And I wonder how AT&T will implement Berkeley UNIX's setitimer() library call? The setitimer() library call is similar to an alarm clock. You can set the timer to "go off" in a specified duration of time. When it goes off, it sends a signal (another method of interprocess communication under UNIX) called SIGALRM. This is used for rough synchronization of processes. AT&T's System V version of setitimer() is called alarm(), which can be set for integer seconds only, due partially to the 1 second maximum time quantum of multi-tasking AT&T UNIX. Berkeley's version can be set for time granularities of 10 microseconds, thanks in part to the hardware of the DEC VAX minicomputer. (Of course on any non-real-time multi-tasking system such as standard UNIX, time can be only approximate.) AT&T certainly does not want any machine dependencies such as a VAX peculiarity in their version of UNIX; after all, isn't portability of UNIX something that Ma Bell wants to hype?

Directory Files Compatible? IBIWISI

What about directories? Directories under AT&T UNIX are arrays of 16-byte structures. The first two bytes are used for the offset index of the file control block (called an "i-node" in UNIX terminology) on the disk. This index is called an "i-number" for inode number. The next fourteen bytes are reserved for the filename; hence the 14 character filename maximum in System V. Berkeley uses a convoluted approach. Directory files are a series of

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

variant sized structures where the filename size, the inumber, and the filename are all squeezed together in a manipulated list. Presumably the Berkeley people wanted to efficiently store the directory entry and at the same time allow the bonus of having filenames of up to 255 characters. If you have programs written assuming the System V view of directories such as

```
struct DIRECTORY_ENTRY
{
     inumber_type    inumber;
     char            filename[14];
} direntry;

    ...later on....

read(fd, &direntry, sizeof direntry);
/* Read one entry from
the directory file */
```

your programs will have a *REAL* hard time reading the Berkeley directories because they are not built that way. (To those System V programmers out there, I have taken the liberty of slightly altering the structure above to increase its clarity for the reader.) The formal structure is declared exactly in /usr/include/sys/dir.h. Instead of reading the directory explicitly, Berkeley programmers have a sensible set of higher level library calls that they can use; e.g., readdir() and scandir().

The $64K Question

Now the question is: How portable is UNIX? Can my programs that rely on Berkeley library routines such as setitimer() be safely ported to AT&T UNIX? How about my System V programs that read the directories via an explicit read() (there are loads of programs that do)? Can I assume that they will work without a hitch on Berkeley UNIX? I really don't think so.

When Icarus Flew Too Close to the Sun, His Wax Wings Melted

As far as I see it, Sun benefits much more than AT&T in this deal. So what if programs written using a common interface definition between the two versions of UNIX will run on an AT&T box? People choose the Sun machine not really because it runs BSD4.2; many other vendors offer 4.2 on their workstations (Silicon Graphics, Apollo, NBI, Masscomp, etc.). They choose the Sun because of its price/performance ratio and because there is so much software written (or being written) for the Sun boxes.

Will the Sun Be Covered by a Big Blue Cloud?

I really think Sun will hold their own against Big Blue and its PC RT. Most of the initial customers for a high-end bit-mapped UNIX-based workstation are either engineers, physical scientists, or other technical types. A prevailing attitude among these people is "get state-of-the-art tools." Did you ever hear a scientist or engineer say, "Now wait a minute! I don't want that FASTER processor or that HIGHER RESOLUTION display or that ELEGANT window system or even that OPEN ARCHITECTURE network file system architecture—I'll play it safe and get an IBM." It's an unlikely scenario to say the least. The time that Sun *really* has to worry is about two years down the road, when these engineering workstations are the corporate workstations (i.e., PCs) of that time.

Sun-day Approaches

By the time you read this, Sun should be a publicly traded company. A recent article in the *Wall Street Journal* states that Sun registered with the SEC for an initial public offering of 4.6 million shares. This will make its four founders instant millionaires.

So even while the rest of the computer industry is in the doldrums, the Sun still shines. The Sun debutante ball should be something to look forward to; after all, it's not just any star; it's the Sun.□

Readers may write to Frank Greco

c/o Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

C DYNAMO

FREE! PCUNIX™

(WITH ANY PURCHASE OF $250 OR MORE. LIMITED TIME OFFER)

LIBRARY: *325 fully tested functions screen handling/graphic, cursor/ keyboard/data entry, string handling, status and control, utility/DOS/BIOS/ time/date, printer control, 4 disks* $129.95

WINDOWS: *The most powerful windowing package available, overlays, borders, pop-up menus/ help windows, zap instantly on/off screen, status lines, horizontal/ vertical scrolling, color control/ highlighting, word-wrap, files to windows, keyboard to windows 3 disks* $129.95

SUPERFONTS FOR C: *Dramatic, high impact screens. Use our character & image libraries and functions, or create your own. Font & Function Library* $49.95

B-TREE LIBRARY: *Fast indexing B-tree. 16 million each: keys and records, unlimited keys per file, variable length records. PC-DOS, UNIX, XENIX, CP/M.* $79.95

ISAM DRIVER: $49.95

COMPATIBILITY: *PC/XT/AT; Full K&R, CI-C86/De Smet/ Lattice/ Microsoft 3.0/Aztec/others.*

VALUE: *All source code, No royalties. Best documentation available.*

No matter what else you have, you need these.

C-TERP: *Simply the best C interpreter available. See C Journal Summer 1985*
C-TERP (specify compiler) $299.95

PCUNIX: *Multi-tasking, networking multi-user. With source code* $99.95
™Wendin Software

COMBINE AND SAVE!

C LIBRARY plus C WINDOWS BOTH for only $179.95
+ SUPERFONTS FOR C . $199.95
+ B-TREE and ISAM . . . $299.95
(A $440 VALUE)
C LIBRARY plus C WINDOWS + SUPERFONTS + C-TERP (a $610 value) $459.95
All (a $740 value) $549.95

Entelekon

12118 Kimberley, Houston, TX 77024
713-468-4412
VISA MASTERCARD CHECK COD

Circle Number 9 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The FASTEST, EASIEST, MOST POWERFUL BASIC EVER!

Amazing New Advance-ments for an Old Friend.

ZBasic is an incredibly advanced and powerful BASIC—but—it's still the old BASIC you're used to. Instead of spending 6 months of your life learning another complicated language, let ZBasic put your programs into light-speed, now! (If you know BASIC, you know ZBasic.)

How Fast is ZBasic?

Lightening fast. Four years of intense development have produced the ultimate BASIC. ZBasic is "Compiled BASIC," and generates stand-alone applications that make any other BASIC completely obsolete. Just look at these speed comparisons:

ZBASIC Compiler	**13.7 seconds**
Turbo Pascal Compiler	**14.1 seconds**
BASIC Interpreter	**2190 seconds**

Sieve from BYTE, Jan., 1983 on an IBM PC, 10 iterations

Compiler Speed/Interpreter Ease.

Like a BASIC interpreter, ZBasic allows you to write and execute your programs immediately! No messy "Linkers," "Loaders," or clumsy "Subroutine Packages" like most other compilers. To compile and edit, simply type "RUN." Debugging works the same as the interpreter, too. Just type "BREAK" or "CTRL C" to get back to the editor.

Lightning-Fast Compilation.

Computer Language Magazine says: "Compilation is amazingly fast." After typing "RUN," ZBasic compiles your program at blinding speed—40 lines per second.

Works the Same on All Computers.

If you're tired of throwing away your old programs everytime you switch to a new computer, ZBasic is for you. Source code is portable from one computer to another, and since ZBasic uses Device Independent Graphics and Disk File commands, your programs automatically "Adapt" to any other computer. And the ZBASIC editor is the same on all versions—regardless of the computer.

Einstein Math.

ZBasic offers programmers a math package that surpasses anything else in the industry! (Yes, ZBasic is even better than FORTRAN, PASCAL, MODULA-2 or any other language available!) You will have up to 54 digits of user selectable accuracy at your power.

"Superb Documentation!"

"The 367 page ZBasic manual is a model of clarity and organization. The documentation is superb, solidifying our impression that someone worked incredibly hard to make ZBasic a benchmark for all other BASIC Compilers."
PC WEEK, Nov. 12, 1985

Easy Structure—If You Want It.

ZBasic helps you "Structure" your programs in a way that's easy and simple...you may use GOSUB or GOTO with names or line numbers. Supports multi-line LONG IFs and LONG FNs. LIST programs with—or without—line numbers! ZBasic automatically indents loops and structures in LISTings, too.

Special MS-DOS Features:

- **Works on IBM PC, XT, AT, JR, TANDY 1000, 1200, 2000, COMPAQ, COMPAQ DESKPRO and most MS-DOS compatible machines. Configures for graphics and text.**
- **Creates stand alone .COM Files**
- **Mouse Support.**
- **Executes DOS Commands, BATCH Programs or other .COM Files from within ZBasic**

"Awesome! It's about time! Great! Unbelievable!"
J.R. CPA
Seymour, MO

"...fast, generates stand alone programs, requires only modest amounts of memory, has outstanding compilation speeds and...was bug free and felt solid. And the price is very attractive."
Bruce W. Tonkin
COMPUTER LANGUAGE

"ZBasic is a powerful offering for BASIC programmers. It provides the flexibility of Turbo Pascal and the speed of compiled BASIC, all at a price that can't be matched. Kudos to Zedcor and to all users who make wise decisions to use ZBasic to the fullest."
Garry Ray
PC WEEK

"The best I have ever seen. I love it! You should be proud of this product."
R. R. Manager
Mesa, AZ

"Mind-blower! Easily the best BASIC I've ever seen!"
J. D.
Baltimore, MD

ONLY ZBASIC GIVES YOU THESE FEATURES:

- **Highlights errors...makes debugging easy!**
- **Not Copy Protected**
- **Never any Royalties or Runtime fees for programs you sell.**
- **Direct commands (Speeds logic testing like an interpreter)**
- **Super Single-Step debug**
- **CHAIN with variable passing. (Share all or some variables)**
- **Create transportable subroutines and functions**
- **Multi-line LONG IF, Multi-line LONG FNs (argument passing)**
- **Decimal, HEX, OCTal or BINary support.**
- **Device-Independent Graphics and File I/O.**
- **Never does String "Garbage Collection"**
- **Comes with "Quick" and "Shell" sort source code**
- **Built in "HELP" screens lets you get answers fast.**
- **Long variable name (15 characters)**
- **Loops: WHILE-WEND, DO-UNTIL, FOR-NEXT-STEP**
- **Serial Port and Modem support**
- **Easily load your old BASIC programs saved in ASCII**

Versions shipping now!

Macintosh, Apple IIe, IIc (128K & DOS 3.3) •
IBM PC and MSDOS 2.1 & Compatibles •
Kaypro Graphics version (CP/M-80) •
CP/M-80 2.0 or greater (Z80 only) •
TRS-80 Model 1/3, Model 4/4p •

$89.95

Send me ZBasic right away! $89.95 complete.

CREDIT CARD—MASTERCARD/VISA/AMEX/C.O.D.

Card Expiration Date: _____

My computer is a _____

Name _____

Address _____ APT. #

City _____ State _____ Zip

Day Phone _____

MAIL TO: ZEDCOR, INC.
4500 E. Speedway, # 93
Tucson, AZ 85712

ZBasic™

Arizona Residents add 5% Sales Tax. In U.S. add $5.00 shipping. C.O.D. add $5.00 per order. Outside U.S. and Canada, add $20.00 per shipment for postage (U.S. currency only.) C, Turbo Pascal, IBM PC, Fortran, Pascal, MODULA COBOL are trademarks of their respective companies—Macintosh is a trademark of Macintosh Laboratories, Inc.

ORDER TOLL-FREE:
1-800-482-4567

30 day money-back guarantee.
Technical Support: (602) 795-3996

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

REUSABLE MODULES

by Paul Barkley

In the back of my head I have this idea that a consultant is supposed to consult—and leave the programming to the programmers. I'm sure that this is the way large consulting firms operate—they can afford to pigeonhole their people. And since the skills needed for these two jobs are not at all the same, specialization makes a great deal of sense.

Unfortunately, we small fry have to wear more than one hat. I have to do the analysis AND the programming, and so does every other consultant that I know. In addition, most of the work I do is on a fixed-fee basis. I tell the client up front how much a job will cost, and then I have to meet that fee agreement. Many other consultants—particularly the sensible ones—won't even consider fixed-fee contracts because of the potential for massive overruns.

Nevertheless, all of us want to do the best quality work that we possibly can, and have to find economical ways to provide sophisticated, bug-free code. Most of the coding I do is in dBASE III, Clipper (a very nice dBASE compiler reviewed in the November/December 1985, issue of *PJ*), and a multi-user database called LAN:DataCore. The latter product is a hard-core programmer's data base designed for local area networks (LANs) and actually consists of a set of routines callable from another high level language, such as FORTRAN, Pascal, C, or Compiled BASIC. I use IBM's Compiled BASIC.

The key to writing very professional code, with a minimum number of bugs, in the shortest amount of time possible, is to create a set of reusable modules. You put the effort into writing the best code that you know how and then use that same code for project after project. As you use it for more projects, you'll think of interesting enhancements that will make it even better. The ultimate goal is a custom program generator —more about that later.

The Input Subroutine

dBASE has a built-in input editor that's invoked whenever you GET and READ. The user has a decent set of commands to edit the contents of one box, as well as another set of commands to move between boxes on the screen. A screen logically has all the information about a given topic—perhaps all the fields in a record, for example, and users really appreciate the fact that they can jump around on the screen to edit previously entered data.

Contrast that with the INPUT statement in BASIC. This statement is fine for beginning programmers who are struggling to understand loops and branches—the least of their concerns is typing in a lot of information. In the real world, though, users such as secretaries and data entry clerks—and the six or seven managers in the world who actually know how to type—all find that programs that use the INPUT statement perform in an unacceptable manner.

Five minutes after installing a new program, you'll find the user's cursor wandering all over the screen as he or she presses cursor commands and types over your nice neat boxes and other prompts. The data that finally gets into the computer is interesting, to say the least. Users just HATE it when they can't correct a mistake on a previous input, and while we programmers never make typos (!), real users make an incredible number.

So, the first reusable module that you need for work in BASIC—or another language that doesn't have full screen editing—is some sort of input routine. The calling program should pass it parameters that define the location and size of the input fields on the screen, as well as any defaults. The results of the input process should be passed back to the program in a standard set of variables. Let the input routine handle type checking on the inputs and refuse to let the user out until he or she has entered a valid date, a number in a number field, or, for example, one of a series of allowable choices. All error messages will be contained in the routine and will appear in a safe part of your screen—often the bottom three or four lines —and will clear themselves.

I use nine different box types, but you may have others that you prefer: integer, real, yes/no (Y/N), date, special matches, alphanumeric, protected, hidden, and password style. Most of these are obvious—I use yes/no questions quite a bit and find this a very useful, if somewhat trivial, type. Protected fields display but don't allow the user to get into them—the cursor will jump over them to the next box. Hidden fields don't display and are useful for internal control numbers where you want

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the contents of a screen edit to go directly into the fields of a record without reassigning variables on a one to one basis. Password fields show the cursor position as you type, but display a blob for the letter pressed.

In BASIC you need to use the INKEY$ statement to get one character at a time, and you'll be responsible for checking it for control codes, cursor movements, and other special keys such as the Insert and Delete. Make your routine handle everything that dBASE can do and then some. I threw in all the WordStar commands as well. Users quickly come to appreciate Control-Y to erase the contents of a hopelessly muddled box and Control-T to erase the next word. Some users prefer to invoke the same commands from the cursor pad—the way the BASIC editor works.

When I used to teach Advanced BASIC, I would require my students to write a simple routine similar to this, but I found that when they were through they didn't have much appreciation for what they had created. You have to write a few programs for other people and watch them crash before the value of this routine becomes apparent. By the way, it runs plenty fast enough in compiled BASIC, although any crazed assembler hackers should feel free to make it really scream in assembler.

Instant-C™:
The *Fastest* Interpreter for C

Runs your programs 50 to 500 times faster than any other C language interpreter.

Any C interpreter can save you compile and link time when developing your programs. But only *Instant-C* saves your time by running your program at compiled-code speed.

Fastest Development. A program that runs in one second when compiled with an optimizing compiler runs in two or three seconds with *Instant-C*. Other interpreters will run the same program in two minutes. Or even ten minutes. Don't trade slow compiling and linking for slow testing and debugging. *Only Instant-C will let you edit, test, and debug at the fastest possible speeds.*

Fastest Testing. *Instant-C* immediately executes any C expression, statement, or function call, and display the results. Learn C, or test your programs faster than ever before.

Fastest Debugging. *Instant-C* gives you the best source-level debugger for C. Single-step by source statement, or set any number of conditional breakpoints throughout your program. Errors always show the source statements involved. Errors always show the source statements involved. Once you find the problem, test the correction in seconds.

Fastest Programming. *Instant-C* can directly generate executable files, supports full K & R standard C, comes with complete library source, and works under PC-DOS, MS-DOS, or CP/M-86. *Instant-C gives you working, well-tested programs faster than any other programming tool.* Satisfaction guaranteed, or your money back in first 31 days. *Instant-C* is $495.

Rational Systems, Inc.

P.O. Box 480
Natick, MA 01760
(617) 653-6194

Circle Number **20** on the Reader Service Card

The Menu Driver

The next module that you should consider is a menu driver. Instead of writing a series of very boring programs that basically put up a menu, accept the user's choice, and then branch to the correct sub-menu or program, you write a single menu program that prints the individual menu items from a list which it obtains elsewhere.

In dBASE, you can easily set up a menu file whose records contain a menu page number, a menu item number, item description, and other information such as report width and appropriate range questions (see the Report Front End). The menu driver starts with a menu page number (say "MM" for the Main Menu) and menu item number (say "01") and proceeds to print that item and each successive item until it hits another menu page. The file, of course, is indexed on page and item. The menus all come out the same, and the user input is always handled in the same way as well. If you preserve the page variable when you chain off to an actual program, you can return to the correct menu page at the conclusion of the program.

When I first heard about menu drivers I thought they were just fantastic—until I tried to use one. Although a menu driver reduces your coding time tremendously and wipes out bugs, the menus themselves simply don't come up on the screen fast enough. It seems to take forever for dBASE to grab each record from the disk, format each line, and print it out. dBASE compilers speed things up, but Clipper won't accept the ambiguity of branching based on the contents of a variable—Clipper wants to know what's connected where at compile time, not run time.

In BASIC, the menu also doesn't run very fast, but there's a simple solution—take all the data out of an indexed file and put it in the program

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

as a series of data statements. It's less elegant, but it runs well. You still have the advantage of being able to easily add new programs to the menus, or rearrange programs into different menus if your users seem to be perpetually lost. By the way, make your menu driver detect programs that aren't yet written and politely tell the user to choose something else. You'll appreciate this yourself as you test the stubbed out menu of a partially completed system.

When you write a new application for someone, all you have to do is change the data statements—the menu driver is completely reusable. Ultimately, I got tired of even the minimal effort required to insert a new data statement and recompile the code, so I wrote one module of a program generator system that uses files to store the menu items and lets you easily change them around, but which then writes the actual lines of compiled BASIC code for the menu driver and inserts the information as data statements (and compiles it again while it's at it). This gives me the advantage of using a file to easily organize the menu structure, coupled with the production speed of data statements. The same thing could be done to write hard coded menus in dBASE and Clipper.

Note that the principle involved, like many good ideas, is pretty simple. Once you've written a menu driver, you obviously know exactly what the code looks like. You then write a program (using BASIC as an example) that simply uses part of your old menu driver as an ASCII sequential input file, reads it one line at a time, and writes it out to the new program (output file). The only interesting part is that your program generator must write out the lines that contain data statements, using data gathered from your indexed file. It has to construct only one or two tricky branching statements along the way, and the result is a custom menu driver that can easily be changed by simply rearranging the menu item data in the menu file and invoking the menu driver generator program. Got it?

Brand New From Peter Norton

A PROGRAMMER'S EDITOR

that's *lightning fast* with the *hot* features programmers need

A WONDERFUL EDITOR FOR PROGRAMMERS FROM PETER NORTON

THE NORTON EDITOR

only $50

Easily customized and saved ▲ *Split-screen editing* ▲ *A wonderful condensed/outline display* ▲ *Structured programming features:* ● *Auto-indenting for* Pascal *and* C ● *A hot "find matching punctuation" feature* ▲ *Great word features too:* ● *Paragraph reformat* ● *Word wrap, word action* ● *You can write your programs* and *your manuals with this one editor* ▲ *Mouse support, too* ▲ *Great for* assembler, *fabulous for* Pascal *and* C

A STEAL AT $50!

Direct from the man who gave you *The Norton Utilities, Inside the IBM PC,* and the *Peter Norton Programmer's Guide.*

"This is the programmer's editor that I wished I'd had when I wrote my *Norton Utilities.* Now I've got it for you to use. You can *program your way to glory* with *The Norton Editor.*"

Peter Norton

Peter Norton, 2210 Wilshire Blvd., #186
Santa Monica, CA 90403, 213-826-8032
Visa, MasterCard and phone orders welcome

Circle 17 on the reader service card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Report Front End

I got tired of writing the same code over and over for each report in a system. It seems that virtually every report needs to know from the user at run time whether to display the information on the screen, print it on the printer, or write it to the disk. In addition, many reports that I write allow the user to choose ranges if they wish. This makes one report do the work of many and functions somewhat similarly to entering restrictive criteria from the command level in dBASE and other data bases. The difference, though, is that the criteria are displayed in English and the user can pick and choose those of interest—the rest are ignored. Our friendly screen oriented input routine is used to display all the criteria at once.

The mechanism for this is the same regardless of the report. If the report is 80 characters or less, it can be displayed on the screen. Since this width of data is contained in one field of the menu driver data, the report front end knows whether to ask the question or not and can redirect the output as required. In addition, I have a file of printer codes on each system so that the report driver can also set the correct typeface and pitch depending on the width of the report and the wishes of the user. This saves a lot of hassle with badly printed reports.

For some reason, one of the most bug-prone areas of report programs is range selection. This is a simple concept—you check each piece of data against the two extremes of the range and either pass it along or skip over it if it flunks the test. What seems to cause the most trouble, though, is accurately matching the user's input values with the actual field data. If you let a blank input signify that the user isn't interested in that end of the range, then you have to find a way to either ignore it later or set it to the first or last value in the file (not easy to do if the index order has nothing to do with the criteria).

I decided to write a range input routine, which is embedded in the report driver. It requests ranges from those criteria applicable to that particular report. The menu file contains a field of check-offs against the list of possible criteria in the system. Typically, it's a field of 20 bytes filled with "Y"s and "N"s. If there's a "Y" in position one, for example, you ask criteria question one and accept input for it. The master list of criteria is unique to the particular application system.

Each report calls a filter subroutine that compares the record against the applicable criteria and returns a status of "OK," "NOK," "BEFORE," or "AFTER." Depending on the report and the data chosen, you might want to use an "AFTER" status to terminate the search through the data. Records certified as "OK" are then processed by the rest of the report.

In addition, I found it useful to create a general report model that I could modify with the word processor. It's important to find a standard method of dealing with multiple breaks, for example, or you'll go nuts trying to debug three or four reports that aren't running correctly, but which you've designed completely differently, using variable names that appealed to you at the time. Write and rewrite your most complicated

ED/ASM-86

The first truly integrated Editor, Assembler, Debugger, and Linker is now available from Oliver Computing Company.

*** The Editor supports block move, block copy, partial save and partial load insertion. All text is tokenized for both efficiency of storage and very fast assembly time. FAST screen I/O and separate editor/debug windows.**

*** The Assembler can be immediately invoked after any editting command. Output can be directed into memory, to a .COM file, .EXE, or to ED/ASM-86's own efficient .LNK file format. 8087/186/286 instructions supported. Macros and most standard pseudo-ops are supported.**
UNIQUE to ED/ASM is its structured assembly language capability. No more wasted labels and time! Multiple segmented programs can be be sent to .COM files.

*** The built in debugger supports symbolic disassembly, integration with the assembler, as well as disassembly of 8087/186/286 instructions. Display 8087 registers in scientific decimal format. SPLIT SCREEN debugging with a dynamic register window is easily invoked. Disassemble programs to disk file for easier analysis. "Single-step" data editting supported for bytes, words, double words, and 8087 floating constants in memory. You can "patch" assemble a line at a time with symbols from previous assembly (even define symbols as you go!)**

*** The linker supports ED/ASM-86 link files, which can be created from virtually any valid ED/ASM-86 source file, even lines with multiple external symbols in expressions. (Note: not compatible with usual link format)**

*** And many more features too extensive to be listed here.**

All of these features exist in one integrated program. A typical development cycle with ED/ASM-86 consists of entering a program with the editor, or editting an existing program, typing "A:,-IM" to assemble into memory, going into the debugger to immediately test the program. Then you go back to the editor for the next cycle.
ED/ASM was developed for Oliver Computing's own use, and has been used for over a year to enhance itself!

ED/ASM-86 is only $95. Send check or M.O. to:
Oliver Computing Company, P.O. Box 90140, Indianapolis, IN 46290 { (317) 849-4450 }
for immediate delivery via UPS.

Circle **15** on the reader service card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

report until you have a system that will handle unlimited breaks. When you discover how to handle the general case, your other specific reports will come much more easily and prove to have fewer bugs.

There are other things that can be done to speed up coding and promote quality, and in the next column I'll talk about the data base driver that I use, and what I consider to be the goal of this quest for reusable modules: a true customizable program generator that writes solid code.

Home Offices Revisited

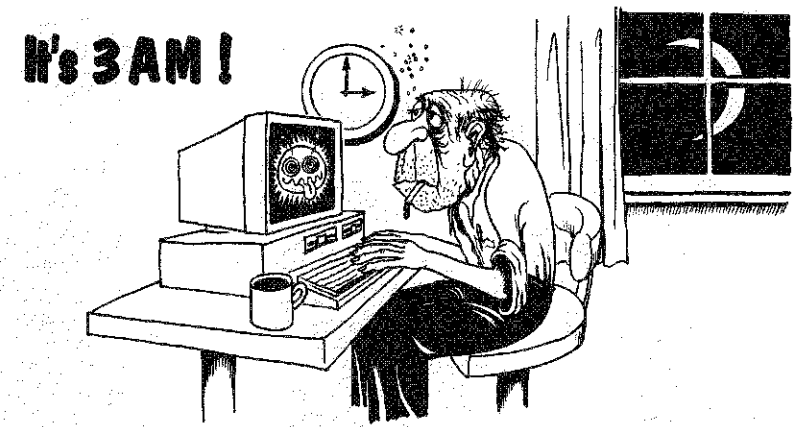
After I wrote my last column on the perils of home offices, a terrible thing happened to me. I found larger space in the same office building for less money, and they would remodel it to suit me. In addition, another tenant badly wanted to expand into my old space and made me an offer, as the godfather used to say, that I couldn't refuse.

Unfortunately, the timing was a little off, and the remodeling went so slowly that I was forced to vacate the old office before the new office was done. I wound up working temporarily from my residence again, and everything I said about the perils of home offices in my last column (*PJ*, March/April 1986) has come back to haunt me. Files are packed in boxes, and software is strewn in several locations. I no longer see clients in my office, of course, because I don't want them wandering around my house. My hardware is jammed together, and all I can think about is when the new office will be done and I can set up everything correctly again.

In Washington, today, the streets are covered in snow and ice, and everyone but me is late for work. Grumpy yuppies are putting dings in their BMWs, getting snow down the collars of their three piece suits, and wrecking their cloth topcoats and leather loafers on their way to the office. I'm comfortable and productive in my jeans and sweater, but of course I don't have to see any clients today either, so I can "hole-up" and write code. Now if I just had enough room for my printer on the desk, and could find the box with the file that I need. . . . Yes, Virginia, home offices CAN be nice.

Drop me a line with your questions, thoughts, or experiences about consulting.□

Paul Barkley
The Barbican Company
6845 Elm Street,
Penthouse One
McLean, Virginia 22101
CompuServe: 74776,25267



Do you know where your bugs are?

This C programmer is finding his bugs the hard way...one at a time.
That's why it's taking so long. But there's an easier way. Use

PC–Lint

PC–Lint* analyzes your C programs (one or many modules) and uncovers glitches, bugs, quirks, and inconsistencies. It will catch subtle errors before they catch you. PC–Lint resembles the Lint that runs on the UNIX* O.S., but with more features and some awareness of the 8086 environment.

- Full K&R C
- Supports Multiple Modules–finds inconsistencies between declarations and use of functions and data across a set of modules comprising a program.
- Compares function arguments with the associated parameters and complains if there is a mismatch or too many or too few arguments.
- User-modifiable library description files for most major compilers.
- All warning and information messages may be turned on and off globally or locally (via command line and comments) so that messages can be tailored to your programming style.
- All command line information can be furnished indirectly via file(s) to automate testing.
- Use it to check existing programs, programs about to be exported or imported, as a preliminary to compilation, or prior to scaling up to a larger memory model.
- All one pass with an integrated preprocessor so it's very fast.
- Has numerous flags to support a wide variety of C's, memory models, and programming styles.
- **Price: $139.00 MC, VISA**
  (Includes shipping and handling) PA residents add 6% sales tax. Outside USA add $10.00
- Runs under MS-DOS* 2.0 and up, with a minimum of 128Kb of memory. It will use all the memory available.

Trademarks: PC–Lint *(Gimpel Software)*, UNIX *AT&T)*, MS-DOS *(Microsoft)*.

NEW!! **Amiga - Lint**
Special Introductory Price
$98.00

GIMPEL SOFTWARE
3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

Circle Number **11** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Tools for the Programmer from Blaise Computing

Save Up To $130 On These Special Offers!

TOOLS & TOOLS 2
For C or Pascal
For a limited time, pick up both packages and save $50 off our regular list price. The C version comes with libraries for the Lattice, Computer Innovations and Microsoft (version 2.03 and 3.00) compilers. The Pascal version supports IBM and Microsoft Pascal. **$175.**

VIEW MANAGER With Source
All libraries are included. Please specify C or Pascal. Regular $425. Save $130. **$295**

Blaise Computing provides a broad range of fine programming tools for Pascal and C programmers, with libraries designed and engineered for the serious software developer. You get clearly written code that's fully commented so that it can serve both as a model and also be easily modified to grow with your changing needs. Our packages are shipped to you complete with comprehensive manuals, sample programs and source code. None of the programs are copy-protected.

FOR C AND PASCAL PROGRAMMERS:

***TOOLS* ◇ $125**
Extensive string and screen handling, graphics interface and easy creation of program interfaces. Includes all source code.

***TOOLS 2* ◇ $100**
Memory management, general program control and DOS file support. Interrupt service routine support. Includes all source code.

***VIEW MANAGER* ◇ $275**
General screen management. Create data entry screens that can be easily manipulated from your application program. Block mode data entry and retrieval with fast screen access.

***VIEW LIBRARY Source* ◇ $150**
Source code to the VIEW MANAGER library functions.

***ASYNCH MANAGER* ◇ $175**
Powerful asynchronous communications library providing interrupt driven support for the COM ports. All source code included.

FOR THE TURBO PASCAL PROGRAMMER:

***Turbo POWER TOOLS* ◇ $99.95**
Extensive string support, extended screen and window management, interrupt service routines, program control and memory management, interrupt filters. All source code included.

***Turbo ASYNCH* ◇ $99.95**
Interrupt driven asynchronous communication support callable from Turbo Pascal. ASYNCH is written in assembler and Turbo Pascal with all source code included.

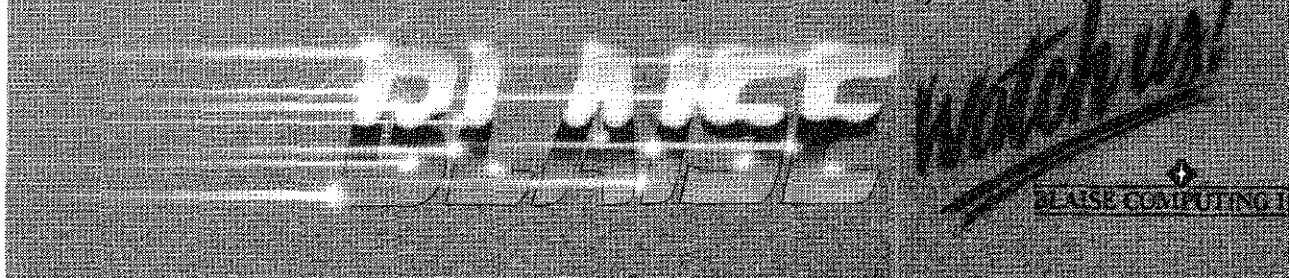
PACKAGES FOR ALL PROGRAMMERS:

***EXEC* ◇ $95**
Program chaining executive. Chain one program from another even if the programs are in different languages. Common data area can be specified. Source code included if you're a registered C TOOLS and C TOOLS 2 user.

***SPARKY* ◇ $75**
Run-time resident (or stand-alone) scientific, fully programmable, reverse polish notation calculator. No limit on stack size, variables or tape. Includes all standard scientific functions and different base arithmetic.

TO ORDER, call Blaise Computing Inc. at (415) 540-5441

♦ 2034 Blake Street ♦ Berkeley, CA 94704 ♦ (415) 540-5441 ♦



Circle Number **2** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

INT24.PAS: A CRITICAL ERROR HANDLER FOR TURBO PASCAL

by Bela Lubkin

Turbo Pascal provides considerable error handling capabilities of its own, but for MS/PC DOS programmers, one important class of error is not covered.

These are the so-called DOS Critical Errors. Unless a program takes specific steps to prevent it, these errors generate messages such as "Not ready error reading drive A Abort, Retry, Ignore?" This not only fouls up carefully laid out data entry screens, but it also gives the user the opportunity to make a wrong decision and wipe out his work.

DOS critical errors are handled by trapping interrupt 24h, DOS Critical Error. Turbo Pascal and its runtime library do not provide a direct way of doing so. This article presents a set of routines to do that trapping and allow the application program to handle the errors as the programmer sees fit.

INT24.INC (Listing 1) consists of four Turbo Pascal procedures and functions. The first procedure, INT24, is the heart of the package. When interrupt 24h is redirected to this procedure, DOS is told how to handle critical errors, and error information is passed back to the Turbo Pascal application.

The second procedure, INT24On, enables that trapping by redirecting interrupt 24h to point to procedure INT24. It also saves the address of the previous interrupt 24h handler and initializes global error condition variables. INT24On should be called once, before any I/O operations in the program.

Procedure INT24Off does the opposite of INT24On. It simply restores the previous interrupt 24h handler. Note that DOS does this automatically when a program terminates; however, if you are running a program interactively within the Turbo Pascal environment, DOS does not consider the termination of *your* program to be a real program termination. It is important to restore the old interrupt 24h handler so that critical errors within the Turbo environment itself will be handled correctly. It is not safe to leave it pointing to INT24 as that code might be overwritten by the environment after the program terminates.

The final routine is function INT24Result. This function is meant to replace Turbo's built-in IOResult function. As with IOResult, calling INT24Result returns a global error code, clearing it in the process. If multiple tests must be done on the error code, assign it to a temporary variable and use this copy of the value. INT24Result calls IOResult and returns its value as part of INT24Result's value. Specifically, the return value of INT24Result is an integer whose low byte is the Turbo Pascal IOResult value, and whose high byte is the DOS critical error number, incremented by one. A value of 0 indicates no error; any other value is an error. The easiest way to use INT24Result is to simply test it for any nonzero value. If more detailed results are needed, the following tables give a complete list of the defined error codes:

Low byte of INT24Result

Turbo 2.0		Turbo 3.0		
Hex	Dec	Hex	Dec	Turbo Pascal error name
00	0	00	0	No error
01	1	90	144	Record length mismatch
02	2	01	1	File does not exist
03	3	F1	241	Directory is full
04	4	FF	255	File disappeared
05	5	02	2	File not open for input
06	6	03	3	File not open for output
07	7	99	153	Unexpected end of file
08	8	F0	240	Disk write error
09	9	10	16	Error in numeric format
0A	10	(99	153)	Unexpected end of file
0B	11	F2	242	File size overflow
0C	12	(99	153)	Unexpected end of file
0D	13	(F0	240)	Disk write error
0E	14	91	145	Seek beyond end of file
0F	15	04	4	File not open

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Fast Path for New Users of Turbo Pascal...

For TURBO PASCAL On The IBM PC

TRAINING WHEELS™

- A comprehensive step-by-step example to help you get started.
- 30 useful & necessary procedures—ready to be included in your program—each with source code and with a simple program illustrating its use: Display help screens & menus quickly, check the DOS environment strings, determine where the program was loaded from, manipulate the archive & read-only bits, determine the ROM date & type of machine & type of display being used, retrieve and parse the command line arguments, make your program preserve the color attributes set for DOS, move a file to a different directory without copying it—and many more.
- Helps get your application going quicker and provides examples of how the DOS function calls are used from Turbo.
- 80-page user-guide plus diskette.

Available for only $25.00
(plus $2 shipping). Cal. Res. add 7%.
Telephone Orders Welcomed (408) 733-4032

LANDMARK SOFTWARE
1142 Pomegranate Court
Sunnyvale, CA 94087

Circle Number **12** on the Reader Service Card

10	16	20	32	Operation not allowed on a logical device
11	17	21	33	Not allowed in direct mode
12	18	22	34	Assign to standard files is not allowed
– –	– –	F3	243	Too many open files

High byte of INT24Result	DOS critical error name [and comments]
0	No error
1	Attempt to write on write protected disk
2	Unknown unit [internal dos error]
3	Drive not ready [drive door open or bad drive]
4	Unknown command [internal dos error]
5	Data error (CRC) [bad sector or drive]
6	Bad request structure length [internal dos error]
7	Seek error [bad disk or drive]
8	Unknown media type [bad disk or drive]
9	Sector not found [bad disk or drive]
10	Printer out of paper [anything the printer might signal]
11	Write fault [character device not ready]
12	Read fault [character device not ready]
13	General failure [several meanings]

Depending on the nature of the error, Turbo Pascal itself may not notice that an error has occurred. In particular, reading from or writing to a DOS character device will seem to succeed, to Turbo's runtime system. INT24Result will show no error in its low byte, the Turbo IOResult value, but the high byte will reflect the proper error. Another problem is that the BIOS on different machines seems to react to character device errors with widely varying speed. Because of these two problems, a write to a character device which is not ready can take a long time; INT24 will be called multiple times, but the Turbo I/O operation will not abort. One solution to this is to write a Turbo user written I/O driver, attached to the device via ConOutPtr, AuxOutPtr, LstOutPtr, or UsrOutPtr. The driver should check the global boolean variable INT24Err and abort if it is true. Here is an example, for writing to the printer:

```
Procedure LstOut(C: Char);
  Var
    Regs: Record Case Integer Of
            1: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: Integer);
            2: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);
          End;
  Begin
    If Not INT24Err Then
      With Regs Do
       Begin
        AH:=5;
        DL:=Ord(Ch);
        MsDos(Regs);
       End;
  End;
```

If the statement

```
LstOutPtr:=Ofs(LstOut);
```

is executed before any printer output, long delays while the BIOS repeatedly fails to output a character will be avoided.

How does it work? The heart of the include file, the procedure INT24, is very simple. It merely sets two global variables that are later interpreted by function INT24Result, then tells DOS to ignore the error. The global variables are INT24Err, a boolean that indicates that a critical error has occurred, and INT24ErrCode, a byte containing the actual error code passed to INT24 by DOS. The variables are initialized by INT24On to indicate no error. INT24Result first calls IOResult, then looks at INT24Err, and if it is true, adds INT24ErrCode as the high byte and returns this value as its result. It also resets INT24Err to indicate no critical error. Thus it clears both Turbo's and its own error conditions.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Advanced use: It may be desirable to ask DOS to retry the error, instead of ignoring it. For this the AL register should be set to 01 instead of 00 in the Inline code in INT24. However, that is too simplistic. An infinite number of retries will do no good unless action is taken to fix the error condition. The INT24 routine must tell the user what to do (e.g., "Please replace disk in drive A!"), then proceed. It may also be useful to give the user the choice, for instance, of retrying or aborting the operation. There is a problem, though. Any code that is called from within INT24 is running inside a DOS function call. Since DOS is not reentrant, DOS function calls cannot be made. Likewise, it is inside a Turbo Pascal runtime library I/O routine; because of that, Turbo I/O calls cannot be made. Any screen I/O done by the INT24 routine must be done through the BIOS, or direct to the hardware. Finally, when INT24 is entered, the data segment register DS will not point to the Turbo program's data segment.

Thus, a modified INT24 procedure that allows the user a choice must do the following: save any registers which will be destroyed by the routine; set DS to an appropriate value if it will be used; use BIOS calls (or direct hardware manipulation—not recommended) to ask the user what to do; restore all registers; pass the appropriate value back to DOS. DOS will respond to the following values in AL:

0 = ignore the error
1 = retry the operation
2 = terminate the program entirely

To abort the current operation, "ignore the error" by passing back a 0 in AL, but set a global condition that notifies the main program that something went wrong. This is the technique used in the standard INT24 routine in the article. To retry the operation, pass back a 1 in AL. Returning a 2 in AL will abort the entire program, a drastic step that may be necessary in some cases.

A good knowledge of assembly language and the low level BIOS or hardware of your machine is necessary to implement these options. Such an implementation is beyond the scope of this article; however, the author and other Turbo Pascal programmers can be reached via the Borland Special Interest Group on CompuServe (GO BOR-100), and this group will probably be able to help you in a more complete implementation.

A small program to exercise the routines, INT24.PAS, is provided in Listing 2. Listing 3, TYPEFILE.PAS, is an even smaller program that demonstrates an "actual" use of the routines.

I hope these routines are useful in *your* applications!□

Bela Lubkin
CompuServe 76703,3015

Code follows.

PRODUCTIVITY UTILITIES
From Opt – Tech Data Processing

NEW **On-Line Help** NEW

A comprehensive utility for adding help windows to your programs. It provides an efficient method for storing help texts in libraries and standard routines for interfacing your programs with these libraries.

Help windows are displayed in a fraction of a second. You have total control over the contents of the window, its position on the screen and the display colors used.

On-Line Help can be interfaced with interpreted Basic and all popular compilers.

On-Line Help includes a Help Library Manager, Library Interface Routines and source code examples. **$149.**

Opt-Tech Sort

High performance sort/merge utility can be used as a stand-alone program or called as a subroutine to many languages.

Supports unlimited filesizes, multiple input files and fixed or variable length records. Many special file types are supported including Btrieve and Dbase. Up to nine sort control fields (ascending or descending), all common data types supported. Output file can be full records, keys or pointers.

Written in assembly language for high performance. Example: 4,000 128 byte records sorted to give key and pointer in 30 seconds. **$99.**

Scroll & Recall

Allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations).

Allows you to easily recall and edit your previously entered DOS commands and data lines without re-typing.

Scroll & Recall is very easy to use. It's a resident utility that's always there when you need it! **$69.**

All programs are fully documented and are compatible with the IBM-PC, XT, AT and compatible machines using the DOS operating system.

Visa, M/C, Check, Money Order, COD, or PO accepted.
Quantity and Dealer discounts available.
To order or to receive additional information write or call:

Opt-Tech Data Processing
P.O. Box 678 — Zephyr Cove, NV 89448
(702) 588-3737

Circle Number **16** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

①

```
Const
  INT24Err: Boolean=False;
  INT24ErrCode: Byte=0;
  OldINT24: Array [1..2] Of Integer=(0,0);

Procedure INT24; { The actual INT 24h handler }
  Begin
    Inline($2E/$C6/$06/ INT24Err /$01/$50/$89/$F8/$2E/$A2/ INT24ErrCode
/$58/
          $8O/$00/$89/$EC/$5D/$CF);
{   Turbo:  PUSH BP                        Save caller's stack frame
            MOV  BP,SP                     Set up this procedure's stack frame
            PUSH BP                        ?
    Inline: MOV  BYTE PTR CS:[INT24Err],1 Set INT24Err to True
            PUSH AX
            MOV  AX,DI                     Get INT 24h error code
            MOV  CS:[INT24ErrCode],AL      Save it in INT24ErrCode
            POP  AX
            MOV  AL,0                      Tell DOS to ignore the error
    .1:     MOV  SP,BP                     Unwind stack frame
            POP  BP
            IRET                           Let DOS handle it from here }
  End;



Procedure INT24On;  { Enable INT 24h trapping }
  Var
    Regs: Record Case Integer Of
            1: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: Integer);
            2: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);
          End;
  Begin
    INT24Err:=False;
    With Regs Do
     Begin
      AX:=$3524;
      MsDos(Regs);
      If (OldINT24[1] Or OldINT24[2])=0 Then
       Begin
        OldINT24[1]:=ES;
        OldINT24[2]:=BX;
       End;
      DS:=CSeg;
      DX:=Ofs(INT24);
      AX:=$2524;
      MsDos(Regs);
     End;
  End;



Procedure INT24Off;

{ Disable INT 24h trapping. This should be done at the end of the program, if
you plan to run the program from within the Turbo compiler. If the INT 24h
handler is left in place, and the compiler gets a critical error, the system is
likely to crash. }

  Var
    Regs: Record Case Integer Of
            1: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: Integer);
            2: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);
          End;
  Begin
    INT24Err:=False;
    If OldINT24[1]<>0 Then
      With Regs Do
       Begin
        DS:=OldINT24[1];
        DX:=OldINT24[2];
        AX:=$2524;
        MsDos(Regs);
       End;
    OldINT24[1]:=0;
    OldINT24[2]:=0;
  End;
```

②

```
Function INT24Result: Integer;

{ INT24Result returns all the regular Turbo IOResult codes if no critical error
has occurred. If a critical error has occurred, then it returns the Turbo
IOResult code in the low byte of the result, and the DOS critical error code in
the high byte.

  If you need the IOResult part, use
   I:=Lo(INT24Result);

  For the INT 24h code, use
   I:=Hi(INT24Result);

INT24Result clears both error codes, so you must assign it to a variable if you
want to extract both codes:

   J:=INT24Result;
   WriteLn('Turbo IOResult  = ',Lo(J));
   WriteLn('DOS INT 24h code = ',Hi(J));

Note that in most cases, errors on character devices (LST and AUX) will not
return an IOResult, only an INT 24h error code. }

  Var
    I:Integer;
  Begin
    I:=IOResult;
    If INT24Err Then
     Begin
      I:=I+Swap(Succ(INT24ErrCode));
      INT24Err:=False;
     End;
    INT24Result:=I;
  End;



Procedure PrinterTest;
  Var LOPTemp: Integer;
  Begin
    WriteLn('Now doing short printer test');
    Write(LST,'T');
    ErrorMessage('Results of printer test:',INT24Result);
    WriteLn('Now doing long printer test with Turbo''s normal LstOutPtr');
    WriteLn('(this may take a long time!)');
    WriteLn(LST,'This is a test');
    ErrorMessage('Results of printer test:',INT24Result);
    WriteLn('Now doing long printer test with custom LstOutPtr');
    LOPTemp:=LstOutPtr;
    LstOutPtr:=Ofs(LstOut);
    WriteLn(LST,'This is a test');
    ErrorMessage('Results of printer test:',INT24Result);
    LstOutPtr:=LOPTemp;
  End;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

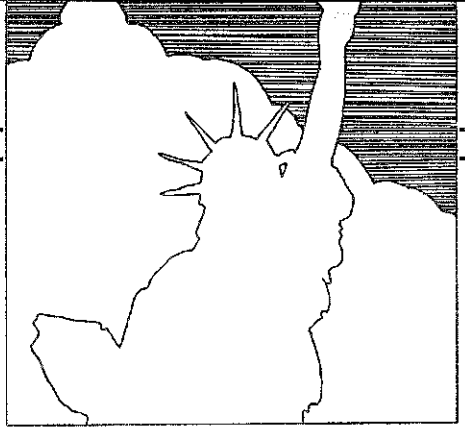
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

③

```
{ This program tests the routines in INT24.INC.
  Run this with printer off (or no printer), and nothing in drive A }

Program INT24Test;

{$IINT24.INC}

Type
  Str255=String[255];

Procedure ErrorMessage(S: Str255; I: Integer);
  Begin
    WriteLn(S,' Error=',I);
    Write('  Turbo 3.0 I/O error=');
    Case Lo(I) Of
        0: WriteLn('No error');
        1: WriteLn('File does not exist');
        2: WriteLn('File not open for input');
        3: WriteLn('File not open for output');
        4: WriteLn('File not open');
       16: WriteLn('Error in numeric format');
       32: WriteLn('Operation not allowed on a logical device');
       33: WriteLn('Not allowed in direct mode');
       34: WriteLn('Assign to standard files is not allowed');
      144: WriteLn('Record length mismatch');
      145: WriteLn('Seek beyond end of file');
      153: WriteLn('Unexpected end of file');
      240: WriteLn('Disk write error');
      241: WriteLn('Directory is full');
      242: WriteLn('File size overflow');
      255: WriteLn('File disappeared');
      Else WriteLn('Unknown error!');
     End;
    Write('  DOS critical error=');
    Case Hi(I) Of
        0: WriteLn('No error');
        1: WriteLn('Attempt to write on write protected disk');
        2: WriteLn('Unknown unit');
        3: WriteLn('Drive not ready');
        4: WriteLn('Unknown command');
        5: WriteLn('Data error (CRC)');
        6: WriteLn('Bad request structure length');
        7: WriteLn('Seek error');
        8: WriteLn('Unknown media type');
        9: WriteLn('Sector not found');
       10: WriteLn('Printer out of paper');
       11: WriteLn('Write fault');
       12: WriteLn('Read fault');
       13: WriteLn('General failure');
     Else WriteLn('unknown error!');
    End;
  End;

Procedure LstOut(C: Char);
  Var
    Regs: Record Case Integer Of
            1: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: Integer);
            2: (AL,AH,BL,BH,CL,CH,DL,DH: Byte);
          End;
  Begin
    If Not INT24Err Then
      With Regs Do
       Begin
        AH:=5;
        DL:=Ord(Ch);
        MsDos(Regs);
       End;
  End;
```

④

```
Procedure FileTest;
  Var
    F: File;
  Begin
    Assign(F,'A:FILE');
    {$I-} Reset(F); {$I+}
    ErrorMessage('Results of disk access test:',INT24Result);
    Close(F);
  End;

Begin
  WriteLn('Make sure that the printer is not ready, and nothing is in drive
A:');
  Write('Hit return to continue: ');
  ReadLn;
  WriteLn;
  INT24On;
  WriteLn('Now trapping DOS critical errors');
  FileTest;
  PrinterTest;
  INT24Off;
  WriteLn;
  WriteLn('Now letting DOS handle DOS critical errors');
  WriteLn('(To proceed with the test, tell DOS to Ignore errors');
  FileTest;
  PrinterTest;
End.




Program TypeFile;

{$IINT24.INC}

Procedure ErrorCheck;
  Var I: Integer;
  Begin
    I:=INT24Result;
    If I<>0 Then
     Begin
      If Hi(I)<>0 Then
        Case Hi(I) Of
          3: WriteLn('Error: no disk in drive!');
          5,7,8,9,13: WriteLn('Error: bad disk!');
          Else WriteLn('Error: DOS critical error #',Hi(I));
        End
      Else
        Case Lo(I) Of
          1: WriteLn('Error: file not found!');
          Else WriteLn('Error: Turbo Pascal I/O error #',Lo(I));
        End;
      Halt;
     End;
  End;

Var
  FileName: String[66];
  InputFile: Text;
  Line: String[255];

Begin
  INT24On;
  Write('Enter the name of the file to type: ');
  ReadLn(FileName);
  Assign(InputFile,FileName);
  {$I-} Reset(InputFile); {$I+}
  ErrorCheck;
  While Not EOF(InputFile) Do
   Begin
    {$I-} ReadLn(InputFile,Line); {$I+}
    ErrorCheck;
    WriteLn(Line);
   End;
  {$I-} Close(InputFile); {$I+}
  ErrorCheck;
  INT24Off;
End.
```



MEMORY MODEL INDEPENDENCE IN ASSEMBLER

by George Defenbaugh

Overview

I was very impressed by Ron Burk's article on 8088 Assembly Macros (Vol. 2, No. 4). [The five macros in Ron's article are used to create functions or procedures and to enter and return from them. —Ed.] Although my subscription started after that issue, I had bought numerous back issues intending to study articles covering assembler concepts. I teach 8088 assembler language and hoped to find code fragments that would help me in my class. Ron's macros are an impressive piece of work and illustrate some of the "deeper" aspects of macro writing. (If you don't have that issue, you should order it.) I now use them as the focal point of the lesson on macros during my class. Ron's article also inspired the development of other macros, and that is the subject of this article.

A friend and I have been developing an assembler library to use when writing programs in C. He has developed most of the library thus far, but as part of that effort I contributed some macros that help to provide memory model independence for assembler subroutines. Although it is a sizable task to write assembler code that is completely memory model and language independent, macros like these can assist the programmer in many ways.

Memory Models Explained

The subject of different memory models might be foreign to programmers who have not studied the architecture of the 8088, or who have used a language which has only one model. The chip was designed with the idea that memory addresses consist of two parts: a segment reference (selector) and an effective address offset. Each of these parts has 16 bits, and depending on the chip, they are combined in different ways.

In the 8088 thru the 8018x, the segment specifies the upper 16 bits of a 20-bit address. The offset is added to the 20-bit address to generate an absolute memory reference. While this sometimes is called a "full" 32-bit address, it is merely the sum of two numbers (the segment and offset) after the segment value has been shifted left four bits. The 32-bit characteristic occurs when passing or storing an address for use elsewhere. In those cases both the segment register (selector) and offset must be passed or stored; hence the need to deal with 32 bits.

This, by the way, clarifies the nature of near and far pointers. A near pointer is 1 word, 16 bits, and indicates the effective address offset of something (either code or data) that utilizes a current segment register (selector) value. A far pointer, however, is 2 words, 32 bits, and requires that a segment register change in the course of using the pointer.

In the 80286 and 80386 chips, the "real" mode matches that described above with some minor differences. The "virtual" mode, however, uses the segment register (selector) as an index into various tables. Those tables contain information pertaining to the actual segment location, its size, security, current presence in memory, etc. Even though the contents of the segment register in virtual mode is very different from the contents when in real mode, it still must be preserved separate from the offset. So we still have 32 bits to maintain even though the segment portion is now a table index rather than an absolute memory reference (hence, the term "selector" to help describe that segment register).

So what does all this have to do with developing assembler subroutines for use with high level languages? Well, suppose that we have an assembler subroutine that we developed for use with one language, say BASIC, and we want to use the same routine as we port that BASIC code to C. (FORTRAN, Pascal, and other language users please bear with us—we'll try to show how this can benefit you too.)

The subroutine for compiled BASIC was a "far" proc. This means that when the subroutine was called, both the segment register (selector) and effective address offset (IP register in this case) were pushed onto the stack. Each of them takes up one word, and the BP register, which is saved upon entry into the subroutine before it is changed, also takes up one word. So there are 3 words, or 6 bytes, between where the BP ends up pointing and the location on the stack where the first piece of data on the stack is found.

Now, however, we come to C and decide to use the small memory model for our program. I use Microsoft 3.0 C, and the memory model terms refer to that compiler. The small model in this case consists of two segments: one for code and the other for data and stack usage; so all code is present in the same segment. The first problem is that because the proc will be "near" to the calling program, the segment register (selector) is NOT pushed onto the stack because it is not changed. This means that there are only 2 words, or 4 bytes, between where the BP register points and any data that was passed. So ALL references in the subroutine to data that was passed must now be adjusted down by 2 and the subroutine re-tested

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

(you WOULD re-test in this situation, wouldn't you?). Then we end up with two pieces of code to care and feed, pieces that do exactly the same thing but must "look" different because of the conventions of the calling program. Then, just as soon as we are getting used to this, we realize that while no one cared what segment name we used for the BASIC subroutine, now we must use __TEXT as the segment name so that the assembled code can be integrated into the compiler output properly—another difference, then, that has to be tested and maintained.

So we dig in and solve the problems of properly addressing the passed data on the stack and correctly referencing the subroutine itself. Then we find that we have not even considered the issues of call by value vs. call by reference, what to do if far pointers vs. near pointers are passed, and how to handle callers who each push arguments on the stack in a different order.

Well, because we are going from BASIC to C, we need to use pointers as arguments so that we match the BASIC convention of passing addresses (references) to data, rather than copies of the data itself. Conversely, when going from C to BASIC we will have problems if we pass a copy of an item rather than its address because BASIC has no feature for utilizing that which is passed as the object itself rather than as the address of the object. And then if a far pointer is passed, we have to utilize the segment register which is part of the far pointer—either that or write the subroutine in such a way that the extra segment (ES) is always used to reference the data. Doing the latter raises the question of where to load the ES from, either from the current DS or from the passed segment register value. Finally, if arguments are pushed onto the stack in another order, well, there's not much you can do except to "reverse" all the references to arguments. References to the third argument would need to become the first, etc. What a pain!

These considerations illustrate that it really is a sizable task to write subroutines that are completely memory model and language independent. The cure can be worse than the disease. But, with judicious use of macros we can eliminate some of the problems. Then we can deal with the more difficult problems on a case by case basis, if at all. This whole situation might dissolve and go away if Microsoft really does follow through with their "common compiler technology" idea. We hope that all subroutine argument lists, in addition to command line prompts and debug procedures, can gravitate toward a common interface.

The Macros Themselves

So we come full circle, back to the original subject of the article. The following set of macros can be used to help (a) set up the offset from the BP register for accessing the first argument on the stack based on the subroutine being near or far, (b) set up each argument's "pseudo" name and offset from the BP register depending on what arguments, if any, occurred before it, (c) clean up the stack by purging the proper number of bytes if that is the responsibility of the subroutine, and (d) set up proc and segment names as required by the chosen model and language. The macros were assembled with Microsoft MASM 3.1. References to BASIC, FORTRAN and Pascal are merely for your information.

The switches (smallc = 0, . . .) select the memory model desired for this assembly. You should choose only ONE model per assembly. You will probably get an error message if you choose more than one. And because you'll be putting the resulting object module in a library of related modules, you'll want to do one assembly per module in order to create the various object modules.

Because the switches have the potential to change each execution while the rest of the macro will remain constant, they probably should be split off and put in a separate include file. A good idea might be to make one include file per model, setting the appropriate switch.

ASMLIB

Assembly Language Programming Library
for the IBM PC/XT/AT or compatible DOS systems.

ASMLIB gives the Assembly Language programmer 190+ assembly functions which do -

- Graphics functions draw CIRCLES, ARCS, ELLIPSES, LINES, and plots POINTS on the Color Adapter, Enhanced Graphic Adapter, and the Herc. Monochrome Card. Functions also allow PAINTING, IMAGE SAVES and RESTORES, and SCROLLING.
- Text Windows - Up to 64 text windows may be defined, outlined, overlapped, moved, and can be grouped onto 256 logical display pages.
- Floating Point - Arithmetic and Trigonometry functions for the MS and IEEE (8087) floating point formats (both 4 and 8 byte precision), and the 8087 (80287) can be utilized automatically if installed into the target system.
- ASCII String/Numeric conversion routines provide a user interface to the math functions. ASFORMAT function allows numeric values to be formatted utilizing commas, dollar signs, left or right justified, etc. (i.e. BASIC's PRINT USING).
- Mouse Support - ASMLIB provides support for any mouse device which adheres to the MS Mouse software standard.
- Dynamic Memory support can utilize all available memory (up to 640k). Blocks of memory can be allocated, changed, moved, and killed.
- Console I/O, Disk I/O with file copy routines, Asynchronous Communications, Printer Support, ASCII String support, Sound generation, plus much more.

ASMLIB is supplied with complete MASM source code on 3 DOS diskettes along with a 215+ page reference manual.

- ALL FOR ONLY -
$149.00

For ordering or info please call:

BC Associates
13073 Springdale St., Suite 134
Westminster, CA 92683
(714) 741-3015

Phone COD orders accepted - ORDER YOURS TODAY!!!

Circle 36 on the reader service card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TURBO SOURCE SEARCH

We are a BBS-by-mail.

Would you like a BBS:

- that AT&T, MCI, SPRINT, et al, won't bill you next month's rent because you needed code?
- that has more than 20MB (and growing) of TURBO pascal source code from which to chose?
- where all files are ARCed allowing 600k+ of code on a 360k disk?

Then join the **SEARCH** today for $25, and receive your first disk containing archive files, our most recent catalog of TURBO code, and the SEARCH *tutorial*.

Select **only** the code **you** want and we custom fill the disks for only $5 per disk. Join now and get 2 FREE disks with your first order!

Join your SYSOP-thru-the-mail, Gerry Power, in the **SEARCH** for the best in public domain and shareware TURBO code.

TURBO SOURCE SEARCH

P.O. Box 876, Scituate, MA 02066, (617) 545-6677

Payment: VISA, MasterCard, Check, COD (disks only)

Circle 35 on the reader service card

Have that include file then include the set of model macros themselves, like this:

```
subroutine:          smallc:              model:
  include smallc       smallc = 0           ...
                       include model        ...
```

The equates for argument sizes define how big each type of passed item is. I show only a few of the items that can be defined. Even though four of the items share two sizes among them, it is clearer and more reliable to define each argument type separately.

The proc__def procedure does two things. First, it insures that the public statement gets done so that an external reference is created. Second, it applies the near or far attribute as necessary. Note that a procedure does not have to be "far" away to require a public statement. There is no proc__end macro because it serves little purpose and provides no convenience. Also, the macro was written this way because MASM does not like you to pass the words near or far as macro arguments. It seems to get very confused.

The segment definition macros apply a particular segment name if needed or concatenate the user supplied name in front of something like __TEXT. Note that the way to get the supplied segment name correctly put on the front of __TEXT and also placed into the ASSUME statement properly is to leave the blank space between the "cs:" and the symbolic "segname." This doesn't cause a problem for the ASSUME statement.

For procs other than the one that is called by the high level language, it might be best to code them explicitly. We'll explain why. They should be "near" to the first proc, and if the macro proc__def is used, then the near/far attribute is supplied based on the memory model chosen. Obviously this might be incorrect if proc__def made the "near" proc into a "far" one. You could use a special version of proc__def, however. Also, the parm__def and return macros expand based on the memory model selected. If largec is chosen, for instance, then the first parm is expected at six bytes off the BP rather than four as it should be. Second, return won't clear the stack (which might be okay, depending on the invoking language, because some languages, such as C, clear the stack themselves upon return).

The end__seg macro is shown even though there is not a corresponding end__proc macro. This is because although procs can have virtually any name, segments often need to have a certain label (__TEXT for instance) and must show it at both the start and end of the segment definition.

Finally we get to parm__def, the "heart" of this whole process. Parm__def begins by setting the counter @parm__nbr to zero, which establishes the initial offset of the first parm depending on the memory model. Note that of the models mentioned, only the small C model allows near calls, and hence near procs. Then irp directive loops once for each parameter in the list <parm__types>. I provided for up to nine parameters only although it can easily be expanded to take more. @parm__nbr counts which iteration we are on.

Then, depending on which iteration we are on, one of the next if statements is done. A "special" symbol, such as parm1@, is set equal to the current bp offset. Then that special symbol is placed inside the bracket expression and added to bp. The resulting expression is assigned to the particular parameter (parm?) using an equate. For instance, <np,i> causes the name parm1 to point to the first parm on the stack <np>, and parm2 points to the second parm <i>. Because np was the keyword for the first parm and we have said that np stands for near pointer and has a size of 1 word, parm2 will point 1 word, or 2 bytes, past where parm1 points. Technically, as Ron pointed out in his article, you are not supposed to redefine the same name with an equate again, but MASM doesn't seem to mind in this case (yet?).

After the equate is done for the current parm value, the offset is bumped depending on the particular parm used. An error check is included to trap the use of symbols that have not been defined. This whole process continues, with one loop of the irp macro occurring for each parameter in the list. The return macro is also listed here because it can utilize the offset generated by parm__def.

It is important to realize that this approach does not allow nested procs or segments. There is no convenient

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

way to save all characteristics of one proc, begin processing another, and then "pop" back to the first. If MASM were to provide an expanded macro processing facility, things like this might become possible. It is also important to realize that once a symbol such as parm1 has been defined, it retains its value until redefined again. The only way to avoid using one of the procs AFTER the proc in which it was defined is to force each symbol value to something like zero each time parm__def executes. That way only the symbols defined this particular time will be usable.

We have spoken of using this approach to simplify language differences. You can also use it with various vendor's compilers of the same language. Each assembly language interface might be slightly different, and these macros can help smooth all that out. These macros have not been "production tested" in a wide range of environments. You may well find some errors, and you'll certainly see room for improvement. I'll be interested in hearing your comments. Thanks in advance. □

References:

Burk, Ron. ASM: 8088 Assembly Macros. *Programmer's Journal*, Vol. 2, No. 4.

iAPX 286 Programmer's Reference Manual. Intel Corporation, 1985. Order number 210498-003.

80386 High Performance Microprocessor with Integrated Memory Management. Intel Corporation, October 1985. Order number 231630-001.

Readers may write to George Defenbaugh at

10034 E. 29th Street
Tulsa, OK 74129-4409

Code follows.

Famous Moments in Computer History #12
Ludwig realizes Martha has invented the first string variable.

IBM PC/AT SYMBOLIC DEBUGGER
"C" FOR YOURSELF!
ONLY CodeSmith -86 has AutoBrowse®
The effortless way to debug your C, PASCAL, & FORTRAN.
Step & Break thru synced SOURCE and Machine CODE WINDOWS.
Dual-mode PATCHING ASSEMBLER
4 STOP-on-Data-Compare conditions
• 256K RAM min.
• DOS 2.0 → 3.x
GUARANTEED 30 DAYS OR YOUR BUCKS BACK
"CodeSmith really shines..." —PC TECH JOURNAL
"It has greatly speeded up the development of new programs and the maintenance of existing programs." —MicroPro Int'l Corp.
"...has saved us uncountable hours in debugging communications software." —Dave Adkins, Jim Dukat, Los Altos CA
"...has almost all the features that are in the Hewlett-Packard 64000 emulator I use at work." —Joel Lagerquist, Riverside CA
"CodeSmith-86...a superb utility." —Steven Kang, New York NY
VISUAL AGE ◆◆ In Calif. (213) 534-4202
642 N. Larchmont Blvd. ★ Los Angeles, CA 90004
ORDER NOW (800) 732-2345
VISA
$145.00

Circle **48** on the reader service card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

①

```
        PAGE    80,132
;-------------------------------------------------------------------------;
;  These macros control the assignment of BP offsets to parameter names.
;  They also help take care of some of the required syntax for subroutine
;  interfacing with various languages.  By doing this, assembler routines
;  can have a standard naming convention to use when addressing parameters
;  on the stack.
;
;  5 parameter types representing pointers (both near and far) and data have
;  been defined.  FORTRAN and PASCAL notes are provided for information only.
;
;               |subroutine|   passed parameters   | parm |  who
;               | location |addresses|     data     | push | cleans
;               |near | far| np | fp | i | l | d  |  seq | stack
;               |-----|----|----|----|---|---|----|------|-------
;  C-small      |  x  |    | x  |    | x | x | x  | r->l | caller
;  C-medium     |     |  x | x  |    | x | x | x  | r->l | caller
;  C-large      |     |  x |    |  x | x | x | x  | r->l | caller
;  QuickBASIC   |     |  x | (1)| (2)|(3)|(3)|(3) | l->r | subroutine
;  FORTRAN      |     |  x | (4)|  x |(3)|(3)|(3) | l->r | subroutine
;  PASCAL       |     |    |    |    |   |   |    |      |
;
;  (1)  When CALL  statement isused
;  (2)  When CALLS statement isused
;  (3)  BASIC and FORTRAN use call by reference ONLY, never call by value
;  (4)  Used when a 4 or 8 byte REAL, or 8 or 16 byte COMPLEX value is
;       returned to FORTRAN caller.
;
;-------------------------------------------------------------------------;
;  George Defenbaugh  10034 E. 29th St.  Tulsa, OK  74129  918-622-7926
;-------------------------------------------------------------------------;

;-------------------------------------------------------------------------;
;  These switches indicate which memory model this library is being
;  assembled to support.  Only ONE of the models can be chosen at a time.
;  A block nesting error will probably result from selecting two switches.
;-------------------------------------------------------------------------;

smallc  =   1
mediumc =   0
largec  =   0
fortran =   0
basic   =   0
pascal  =   0

;-------------------------------------------------------------------------;
;  These equates define the size of each argument type which can be passed
;-------------------------------------------------------------------------;
;                   argument keyword      description
NP      equ  2  ;      np                 2 byte pointer (near pointer)
FP      equ  4  ;      fp                 4 byte pointer (far pointer)
I       equ  2  ;      i                  integer, 2 bytes
L       equ  4  ;      l                  long integer, 4 bytes
D       equ  8  ;      d                  double precision, 8 bytes

;-------------------------------------------------------------------------;
;  This macro controls the definition of a proc.  The near or far attribute
;  of the proc statement is controlled by the memory model set above.  This
;  proc is to be used to define each proc in this library.  There is no
;  corresponding macro to control the definition of the end of the proc.
;  It takes as much code to define the end of a proc explicitly as it does
;  to use a macro, and the name of the proc has to be coded either way.
;  There is nothing special about the name of a proc, unless you count that
;  languages like C sometimes require things like a leading underscore, etc.
;-------------------------------------------------------------------------;

proc_def        macro       proc_name

                public      proc_name       ;;generate an external reference
if              smallc
proc_name       proc        near
else
proc_name       proc        far
endif

                endm                        ;;end proc_def macro
```

②

```
;-------------------------------------------------------------------------;
;  These macros control the definition of segments.   For some memory
;  models the name of certain segments must be a particular value.  For other
;  models any segment name can be used.  Segments are like procs in that
;  the segment name used when the segment is defined must also be used when
;  it is ended.  A macro to end the segment is used here because for some
;  models the segment name is 'hidden' from the caller.  This same concept
;  can be applied to data segments too.
;-------------------------------------------------------------------------;
code_seg_def    macro       segname

if              smallc
_TEXT           segment     byte public 'CODE'
                assume      cs:_TEXT
endif

if              largec
segname&_TEXT    segment
                assume      cs: segname&_TEXT
endif

if              basic
code_seg        segment
                assume      cs:code_seg
endif

                endm                        ;;end code_seg_def macro

end_code_seg    macro       segname

if              smallc
_TEXT           ends
endif

if              largec
segname&_TEXT    ends
endif

if              basic
code_seg        ends
endif

                endm                        ;;end end_code_seg macro
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

③

```
;-------------------------------------------------------------------------------;
;  This macro assigns the names parm1, parm2, ... to arguments found on the     ;
;  stack.  It will appear at the front of the proc, and will determine the      ;
;  size of each argument based on a keyword found in the parm list              ;
;  associated with the proc.  The size of each parm might be different,         ;
;  (not all parms passed to a proc are one word long), so the positive offset   ;
;  from the BP register has to be calculated by examining each keyword.         ;
;-------------------------------------------------------------------------------;
parm_def        macro       parm_types

@parm_nbr       =           0

if  smallc
    @parm_offset    =       4
else
    @parm_offset    =       6
endif

irp         parm,<parm_types>
    @parm_nbr       =       @parm_nbr + 1

    ife @parm_nbr - 1
        parm1@      =       @parm_offset
        parm1       equ     [bp + parm1@]
    endif

    ife @parm_nbr - 2
        parm2@      =       @parm_offset
        parm2       equ     [bp + parm2@]
    endif

    ife @parm_nbr - 3
        parm3@      =       @parm_offset
        parm3       equ     [bp + parm3@]
    endif

    ife @parm_nbr - 4
        parm4@      =       @parm_offset
        parm4       equ     [bp + parm4@]
    endif

    ife @parm_nbr - 5
        parm5@      =       @parm_offset
        parm5       equ     [bp + parm5@]
    endif

    ife @parm_nbr - 6
        parm6@      =       @parm_offset
        parm6       equ     [bp + parm6@]
    endif

    ife @parm_nbr - 7
        parm7@      =       @parm_offset
        parm7       equ     [bp + parm7@]
    endif

    ife @parm_nbr - 8
        parm8@      =       @parm_offset
        parm8       equ     [bp + parm8@]
    endif

    ife @parm_nbr - 9
        parm9@      =       @parm_offset
        parm9       equ     [bp + parm9@]
    endif

    ifidn       <parm>,<np>
        @parm_offset    =       @parm_offset + NP
    else
        ifidn       <parm>,<fp>
            @parm_offset    =       @parm_offset + FP
        else
            ifidn       <parm>,<i>
                @parm_offset    =       @parm_offset + I
            else
                ifidn       <parm>,<l>
                    @parm_offset    =       @parm_offset + L
                else
```

④

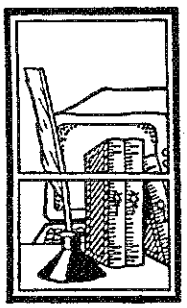
```
                ifidn       <parm>,<d>
                    @parm_offset    =       @parm_offset + D
                else
                    %out invalid parameter parm found in parm_def expansion
                endif
            endif
        endif
    endif
endm
            endm            ;;end parm_def macro

return      macro

if  smallc
        ret
endif

if  mediumc
        ret
endif

if  largec
        ret
endif

if basic
        ret     @parm_offset        ;;subroutine cleans up stack
endif

            endm            ;;end return macro
```

```
        include   model.mac
;-------------------------------------------------------------------------------;
; Test code for MODEL macros.  Note that if you want to write a proc which is
; near to this code, then you should just code it without use of the macros.
; Its location from here will not change due to the memory model.
; This code is not meant to accomplish anything except to exercise the
; macros.
;
        code_seg_def   lc

        proc_def  proc_one
;
        parm_def  <np,i>
        mov       si,parm1
        mov       di,parm2
        call      proc_two
        return
;
proc_one            endp
;
proc_two proc       near                ;proc done without proc_def macro
;                                       ;  shows that parm_def works okay
        parm_def  <fp,l,d,x>            ;x illustrates the error message
        mov       si,parm1
        mov       si,parm2
        mov       si,parm3
        ret
;
proc_two            endp

        end_code_seg   lc

        end
```

AN APPROACH TO WINDOWING FOR THE ASSEMBLY LANGUAGE PROGRAMMER

by C. D. Hawkinson

BC Associates
13073 Springdale St.
Suite 134
Westminster, CA 92683
(714) 741-3015



Chris Hawkinson is the author of ASMLIB, an assembly language function library.

As a professional programmer proficient in many programming languages (C, Pascal, FORTRAN, COBAL, BASIC, etc.), I have come to realize the many programming advantages that assembly language offers: among them, flexibility, total access to computer resources, program efficiency, small size, and fast execution.

But there is a major drawback to assembly language programming—no access to high level functions. When I need to write a graphics-related program, windowing, or a math module, I must code that portion in BASIC or C and then link in assembly language routines to help speed up my program. The end result is a program that is much larger and slower than it needs to be. But how else could I write my program?

This problem led me to write ASMLIB, an MS Assembly Language library package for the IBM PC XT or AT that gives the assembly language programmer access to many of the functions found in BASIC or C.

The library now contains 190 functions, including Floating Point math routines (IEEE or MS), automatic 8087 support, Graphics routines (Circles, Arcs, Lines, Paint, etc.), Windowing functions (Define windows, Color windows, Move windows, etc.), and Dynamic memory handling (up to 640K can be handled directly).

Windowing

The windowing routines handle up to 64 active display windows on 256 logical display pages. A window may be of any size up to 256 columns by 256 rows and have any display/color attribute assigned to its display area.

Windows may also overlap one another or be partially off the actual display screen.

The windowing functions fall into three catagories:

1. WINDOW DEFINITION: The definition functions intialize a new window, modify an existing window, or delete a window.

The definition function, __UWINDOW__DEF, is passed with information about the window's size (X,Y coordinates of upper-left and lower-right corners), the window's logical display page (a window may be placed onto one of 256 different logical pages), the window's display attribute (color, blinking, reversed, etc.), and the window's buffer location (the calling program must maintain the buffer areas for each window). __UWINDOW__DEF clears out the window's buffer area, draws the window's outline on the display, and then returns a 16-bit ID code to be used for any calls referencing this window.

A window's size, location, display page, or color will often need to be changed. To do this, a program doesn't need to delete the old window and then redefine it with the definition function. __UWINDOW__CHANGE modifies the window's parameters and then redisplays the window. __UWINDOW__CHANGE is called with the same entry parameters as the definition function.

When the program wishes to remove a window, it must call __UWINDOW__DEL, which deletes the specified window from the display area. The window's ID can then be assigned to another window through the definition function, above.

2. WINDOW HANDLING: These functions allow windows to be selectively displayed in pages or individually.

When defined, each window is mapped onto a logical display page (1 of 256). The windowing portion of ASMLIB will display only windows that are on the current active display page (defaults to page 0); all other windows will be held in memory, but will not be placed on the display screen. By calling __UWINDOW__PAGE, the applications program can indicate the logical page to use for display. This allows some windows to be displayed while others may be getting modified in memory; then the new page can be "switched" onto the display screen.

Once the display page has been set, the windows will be placed onto the screen in their defined locations. Because windows may overlap each other there must be a way to define which window should be placed on top of the heap. This capability is provided by the __UWINDOW__TOP function. The program passes a window ID to the function, and the function then "floats" this window to the top of the display window "heap." Once topped, this window will always be displayed in its entirety, with all other windows displayed beneath it.

Any window, whether topped or not, may have data written to it or read from it without disturbing the other windows. The top function modifies only how a window is displayed.

3. WINDOW I/O: The I/O functions allow data to be written to any display window or read from any window. Display windows may be cleared or scrolled up, down, left, or right. The window's cursor may also be directly addressed through a library function.

The __UWINDOW__CHR and __UWINDOW__STG

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

functions write either one byte or an entire string (null terminated) of data to the display window. These routines treat all characters except Line Feed (0Ah) and Carriage Return (0Dh) as literal bytes of data that display one character on the console and move the cursor forward one position. The LF and CR characters perform their normal functions by moving the cursor down one line or to the leftmost position. The window's cursor is automatically moved after each byte is displayed, but the window is not scrolled automatically. Instead, the cursor just wraps back around to the window's top left corner.

The __UWINDOW__CLEAR function call clears the window's display area to spaces and then homes the window's cursor.

The window's cursor position may be controlled via the __UWINDOW__CUR function. All cursor positions are expressed in relative terms (i.e., coordinates 0,0 will always be the home position in the window) from 0,0 to the window's maximum size, which was set when the window was defined initially.

Scrolling of the window is performed by the __UWINDOW__SCROLL function call. This one function can scroll a window up, down, left, or right. Scrolling can be done one line (or column) at a time, or it can be done any number of times (up to 65,535) with one function call.

__UWindow__DEF

Following is a listing of the __UWINDOW__DEF function.

First, the function saves all entry registers, then places the definition parameters on the stack. The subroutine __Check__Empty__Table is called to find an entry in the window list table which is not in use.

If an unused entry can't be found, execution jumps to the error routine to exit the program.

Otherwise, the buffer pointer's address is popped off the stack and placed directly into the table entry. The window's page and attribute bytes are popped off the stack and placed into the table, and the page value is also placed into a local variable for later use.

Next, the window's upper-left and lower-right coordinates are popped off the stack. The coordinates are tested to be sure the upper-left values are for the upper-left corner and the lower-right values for the lower-right corner.

The proper locations in the table entry are updated, and the cursor position is also set. The __Buf__Size subroutine is called to calculate the total byte size of the window's display buffer; then this value is saved into the table. The buffer's pointer address is pulled out of the table entry; then the actual buffer address is read from the pointer, and the entire buffer is space filled. The __Top subroutine is called with our new window ID in the AX register. This subroutine draws the new window's outline and displays its contents at the top of the display "heap." The function finishes by popping all entry registers off the stack and returning to the calling program.

The sample program first defines three display windows. The third display window is now on top of the "heap" and will be viewed in its entirety. The other windows will be partially obscured by the windows on top of them. The program forces the second window to "float" to the top and then waits for a string to be entered via the keyboard. This string is echoed to the second window (on top) and also to the first display window (on bottom). Of course, the data placed into the first window will not be totally visible due to the two overlapping windows. After the return key is pressed, the first window is "floated" to the top to display the data that was echoed to it. Then the program clears the second window and exits back to DOS.□

```
;---------------------------------------------------------------------
; _UWINDOW_DEF - Define a new window. The new window will always
;       be at the top of the 'heap.'
;
;       In - AH = display page for the new window.
;            AL = display attribute to use in window.
;            BH,BL = Upper left corner of window.
;            DH,DL = Lower right corner of window.
;            ES:DI must point to a pointer to the buffer area for
;               the window.
;       Out - CF set if error.
;             AX = ID for the new window.
;---------------------------------------------------------------------
CProc   _UWINDOW_DEF

        Push    Bx                      ;Save the entry registers.
        Push    Cx
        Push    Dx
        Push    Si
        Push    Di
        Push    Ds
        Push    Es
        Push    Bp

        Push    Bx                      ;Save the upper left coords.
        Push    Dx                      ;Save the lower right coords.
        Push    Ax                      ;Save the page and atb bytes.
        Push    Di                      ;Save the buffer pointer address.
        Push    Es
        Call    __Check_Empty_Table     ;Find an empty table entry.
        Jc      _UWINDOW_DEF_Err4       ;Error if we don't have a table entry.
        Pop     Es:[Di][WBLK_SEG]
        Pop     Es:[Di][WBLK_OFF]       ;Set the buffer pointer address.
        Pop     Bx
        Mov     Es:[Di][WBLK_PAGE],BH   ;Save the display page for this window.
        Mov     Es:[Di][WBLK_ATB],BL
        Mov     Cs:[__UWATB],BL
        Pop     Dx
        Pop     Bx

        Cmp     Dh,Bh                   ;Dh > Bh ?
        Jnb     _UWINDOW_DEF0           ;Yes, so jump.
        Xchg    Dh,Bh
_UWINDOW_DEF0:
        Cmp     DL,BL                   ;DL > BL ?
        Jnb     _UWINDOW_DEF1           ;Yes, so jump.
        Xchg    DL,BL
_UWINDOW_DEF1:
        Mov     Es:[Di][WBLK_TROW],BH   ;Save the top row.
        Mov     Es:[Di][WBLK_CROW],BH
        Mov     Es:[Di][WBLK_BROW],DH   ;Save the bottom row.
        Mov     Es:[Di][WBLK_LCOL],BL   ;Save the left column.
        Mov     Es:[Di][WBLK_CCOL],BL
        Mov     Es:[Di][WBLK_RCOL],DL   ;Save the right column.
        Push    Ax                      ;Save the ID.
        Call    __Buf_Size
        Mov     Es:[Di][WBLK_BSIZ],Cx   ;Save the buffer size.
        Sub     AL,AL
        Les     Di,Es:[Di][WBLK_OFF]    ;Get buff pointer address.
        Les     Di,Es:[Di]              ;ES:DI points to screen buffer area.
        Mov     AL,' '                  ;Space fill buffer.
        Cld
        Rep     Stosb                   ;Fill it all.
        Pop     Ax                      ;Get our id back.
        Push    Ax
        Call    __Top                   ;Draw this window on top.
        Pop     Ax
        Clc
_UWINDOW_DEF_DNE:
        Pop     Bp                      ;Restore entry registers.
        Pop     Es
        Pop     Ds
        Pop     Di
        Pop     Si
        Pop     Dx
        Pop     Cx
        Pop     Bx
        Ret

_UWINDOW_DEF_ERR4:
        Stc
        Pop     Ax                      ;Clean up our stack.
        Pop     Ax
        Pop     Ax
        Pop     Ax
        Jmp short _UWINDOW_DEF_DNE

_UWINDOW_DEF    Endp
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Vitamin C™
A Healthy Supplement For C Programmers

Perfect Windows, Powerful Data Entry, FULLY Integrated, For Effortless Data Entry Windows!

- Complete input formatting
- Unlimited Validation
- Full attribute control
- Field sensitive application help system
- Multiple virtual windows
- Fully automatic, collision proof overlay and restore
- Print to & scroll background windows
- Animated window "zoom"
- Move, grow, shrink, hide, or show any window
- "Loop function" allows processing while awaiting input

and much much more!

Designed to increase your productivity and help you produce superior applications in dramatically less time! Includes 100% source, tutorial, reference manual, examples, and sample programs.

Specify Microsoft, Lattice, Computer Innovations, Aztec, or Mark Williams. Ask about Unix.

$149.95

NOW... VCScreen !

Our new interactive screen "painter" actually lets you draw your data entry windows! Define fields, text, boxes & borders. Move them around. Change attributes. Then the touch of a button generates C source code calls to the Vitamin C routines! Now, your screen designs will be faster and more pleasing when they are created with VCScreen!

Requires Vitamin C library above. For IBM & compatibles. **$99.95**

For Orders Or Information,

(214)245-6090

Creative Programming Consultants

Box 112097 Carrollton, TX 75011-2097

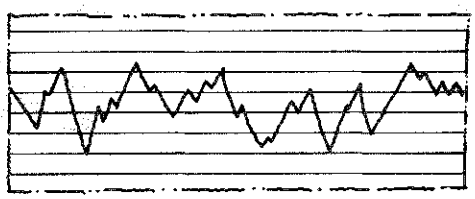
Include $3 ground, $6 air, $15 overnight shipping, $25 if outside USA. Texans add 6% tax. All funds must be in U.S. dollars drawn on a U.S. bank.

Circle **40** on the reader service card


```
;=============================================================
; WINDOWS using ASMLIB - Sample program
;=============================================================
        Extrn   _UWINDOW_DEF:far,_UWINDOW_DEL:far,_UWINDOW_TOP:far
        Extrn   _UWINDOW_CHR:far,_DKEY_INPUT:far

Code    Segment Para    Public  'Code'
        Assume  Cs:Code
        Assume  Ds:Data
        Assume  Es:Data

Window_Test Proc        Far

        Push    Ds                      ;Save the DOS return address.
        Sub     Ax,Ax
        Push    Ax

        Mov     Ax,Data                 ;Set up our data and extra segments.
        Mov     Ds,Ax
        Mov     Es,Ax

;1) Define 3 windows.

        Mov     Ah,0                    ;Set this window's page to 0 (default).
        Mov     AL,3                    ;We'll make the chrs cyan on black.
        Mov     Bh,1                    ;Set the upper-right to 1,1.
        Mov     BL,1
        Mov     Dh,15                   ;Set the lower-left to 15,60.
        Mov     DL,60
        Mov     Di,Offset Window_1_Pnt  ;Point to the buffer pointer.
        Call    _UWINDOW_DEF            ;Define our window.
        Jc      Window_test_error       ;Jump out now if we error'd.
        Mov     [Window_1_ID],Ax        ;Save this window's ID value.

        Mov     Ah,0                    ;Set this window's page to 0 also.
        Mov     AL,2                    ;We'll make the chrs green on black.
        Mov     Bh,10                   ;Set the upper-right to 10,12.
        Mov     BL,12
        Mov     Dh,20                   ;Set the lower-left to 20,50.
        Mov     DL,50
        Mov     Di,Offset Window_2_Pnt  ;Point to the buffer pointer.
        Call    _UWINDOW_DEF            ;Define our window.
        Jc      Window_test_error       ;Jump out now if we error'd.
        Mov     [Window_2_ID],Ax        ;Save this window's ID.

        Mov     Ah,0                    ;Set this window's page to 0 also.
        Mov     AL,6                    ;We'll make the chrs brown on black.
        Mov     Bh,5                    ;Set the upper-right to 5,20.
        Mov     BL,20
        Mov     Dh,10                   ;Set the lower-left to 10,75.
        Mov     DL,75
        Mov     Di,Offset Window_3_Pnt  ;Point to the buffer pointer.
        Call    _UWINDOW_DEF            ;Define our window.
        Jc      Window_test_error       ;Jump out now if we error'd.
        Mov     [Window_3_ID],Ax        ;Save this window's ID.

; See figure 3.
; TOP the 2nd window.

        Mov     Ax,[Window_2_ID]        ;Get the 2nd window's ID.
        Call    _UWINDOW_TOP            ;Force this window to the top.

; See figure 4.
; Input from keyboard, and echo to the window's.
Window_Test_Loop:
        Call    _KEY_INPUT              ;Wait for a key to be pressed.
        Cmp     AL,0Dh                  ;Is it a carriage return ?
        Je      Window_Test_Go_On       ;Yes, so jump to continue.
        Mov     DL,AL                   ;DL = character read in.
        Mov     Ax,[Window_2_ID]
        Call    _UWINDOW_CHR            ;Display one char on window #2.
        Mov     Ax,[Window_1_ID]
        Call    _UWINDOW_CHR            ;Also on window #1.
        Jmp short Window_Test_Loop      ;Loop back for more characters.
Window_Test_Go_On:

; Float the 1st window.
        Mov     Ax,[Window_1_ID]        ;Get the 1st window's ID value.
        Call    _UWINDOW_TOP            ;Float this one to the top.

; Clear the 2nd window.
        Mov     Ax,[Window_2_ID]        ;Get the 2nd window's ID.
        Call    _UWINDOW_CLEAR

;Now back to DOS.
Window_Test_Error:
        Ret                             ;Return to DOS.

Window_Test     Endp
Code    Ends

Data    Segment Para    Public  'Data'

Window_1_ID     Dw      0               ;These are the window ID variables.
Window_2_ID     Dw      0
Window_3_ID     Dw      0

Window_1_Pnt    Dd      Window_1_Buff   ;These are the window buffer pointers.
Window_2_Pnt    Dd      Window_2_Buff
Window_3_Pnt    Dd      Window_3_Buff

Window_1_Buff   Db      15*60 Dup(?)    ;First window has dim 15 by 60.
Window_2_Buff   Db      11*39 Dup(?)    ;Second window has dim 11 by 39.
Window_3_Buff   Db      6*56 Dup(?)     ;Third window has dim 6 by 56.

Data    Ends
        END
```

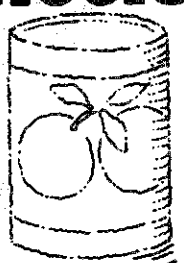

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

VEDIT® Plus Text Editor



The Navy charts new concepts with it... GM engineers the future with

it... National Can preserves facts



with it...GE has bright ideas with it... Here's why you shouldn't be without it.

Every day, VEDIT PLUS helps thousands of programmers, writers and engineers get down to business.

So why do people who could have ANY text editor prefer ours? For a lot of reasons, including:

- CAPACITY—With VEDIT PLUS, file size is *never a problem. And virtual disk buffering* simplifies editing of even the largest files.
- FLEXIBILITY—VEDIT PLUS lets you edit up to 37 files simultaneously. So you can cut and paste. Edit programs. Edit text. Even perform numerous search/replace functions in several files without user intervention.*
- CUSTOMIZATION—With VEDIT PLUS, you can create your own on-line editing functions with keystroke macros. Develop your own on-line help screens. Determine and revise your own keyboard layout easily.
- SPEED—VEDIT PLUS not only works hard, it works fast. Faster, in fact, than any other text editor on the market.
- EXPERIENCE—Six years ago, CompuView revolutionized the concept of microcomputer text editing. And we've been improving our products and services ever since.

Special Offer: Order a VEDIT PLUS text editor for $225 and we'll include our V-PRINT™ document formatter—a $120 value—absolutely free.

Call CompuView today at 313/996-1299. You'll be in good company.

CompuView®

CompuView® Products Inc., 1955 Pauline Boulevard—Suite 300, Ann Arbor, Michigan 48103, TELEX 701821

Available for PC DOS, MS-DOS, CP/M, CP/M-86.

*Free sort, compare, print and main menu macros included; optional 8080-8086 translator or mailmerge, $50 each.

Circle 39 on the reader service card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE PROPOSED ANSI C LANGUAGE STANDARD, PART 2

by Rex Jaeschke

© *Rex Jaeschke, 1986. All rights reserved.*

A number of MS/PC DOS C compiler vendors have released new versions that implement some of the new capabilities in the proposed ANSI C Language Standard. These vendors include Microsoft (V3), Lattice, Eco-soft, and Whitesmiths. Wizard's new version should also be available by the time you read this.

High C, debuting from MetaWare of Santa Cruz, CA, (408) 429-META, merits special mention. While MetaWare is a new player in the DOS marketplace, they are seasoned veterans of the compiler business, having supplied translation writing systems to many and well known software and hardware vendors for a number of years. The principals of MetaWare are respected grammarians Frank DeRemer and Tom Pennello.

High C is no toy: It is a serious and powerful product for serious users. MetaWare claims that High C supports EVERYTHING in the proposed Standard except the new keywords const and volatile; and that's saying something. High C also has a number of extensions and includes a number of useful UNIX-like utilities. Distributed on seven diskettes, High C supports five models. The compiler itself, HC.EXE, is approximately 500Kb (yes, you read right) alone, and MetaWare had to write a special utility so that the file could be split over two diskettes for distribution. Obviously, a hard disk is necessary for this product. Despite its size, High C is lightning fast and has a cross-reference option and good diagnostics. It also has an ansi switch so that extensions can be flagged. If you want a preview of the proposed ANSI Standard, High C is the compiler to buy. It's slick.

The Preprocessor

A directive may optionally have white space before the # character and between the # and the directive name. This implies that a directive need not begin in column 1 of a source line. (White space, in this context, is limited to spaces and horizontal tabs.)

Conditional Inclusion

The #if directive may contain constant expressions. All arithmetic is done as if the operands were of type long int in the translation environment. The execution environment (which may be different from that of translation) is of no consequence here. Whether a single-character character constant can have a negative value is implementation-defined. The expression may not contain the size of operator, a cast, or an enumeration constant; hence, the following common practice is no longer sanctioned:

```
#if sizeof(int) == 2
        long count;
#else
        int count;
#endif
```

A new preprocessor-only operator has been added; namely, defined. It is used as follows:

```
#if defined(8086L) && !defined(DEBUG)
        ...
#else
        ...
#endif
```

This operator does not provide any new capability; it simply makes compound expressions easier to write and read. The "old" method for the example above would be:

```
#ifdef 8086L
#ifndef DEBUG
        ...
#endif
#else
        ...
#endif
```

The defined operator may be written as

```
defined(identifier)
```

or

```
defined identifier
```

Like the defined operator, the #elif directive has been added to make complex expressions easier to write and read. For example,

```
#if condition-1
        ...
#elif condition-2
        ...
#endif
```

could have been written as

```
#if condition-1
        ...
#else
#if condition-2
        ...
#endif
#endif
```

If an identifier's value is tested and that identifier has not been defined, it defaults to a value of zero (or false); hence, it is not necessary to test for the existence of an identifier before using it—provided, of course, its absence is to be treated just as if it were defined with a value of zero.

Despite strong sentiment from several quarters to add some kind of checkable identifier to the #else and #endif so that the compiler could match #if's with #endif's, no new syntax was added for this purpose; consequently, no

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

token may be appended to lines containing these directives although (as always) a comment is permitted.

The Standard guarantees at least six nesting levels for conditional compilation.

The #include Directive

A new #include directive format has been added as follows:

```
#define FILENAME "file.dat"
#include FILENAME
```

This example is trivial although the concept may be useful, particularly if the macro FILENAME is defined on the compiler's command-line. The operand used is an identifier and may not necessarily be a preprocessor macro. For example, DEC has implemented this capability on VAX-C, which runs on the VAX-11 series under VAX/VMS, in a way that such identifiers are considered to be entry points into a text library; that is, headers need not be stored in individual disk files; they could be objects within a library of some kind.

With the new preprocessor "paste" operator and string concatenation it is also possible to construct a macro definition such as a filename. This would allow the device and/or directory part to be passed via the compiler command-line, and the filename to be hard-coded as follows:

```
#define FILENAME "##DEVICE##DIRECT##file.dat"
#include FILENAME
```

If DEVICE and DIRECT were passed as command-line arguments of c: and \dir\ respectively, the example would become

```
#define FILENAME "c:\dir\file.dat"
#include FILENAME
```

The new preprocessor-only concatenation operator *##* allows tokens to be pasted together to form new tokens as shown. The *##* can optionally be surrounded by spaces.

The same result can be achieved using string concatenation as follows:

```
#include FILENAME DEVICE DIRECT "file.dat"
```

provided DEVICE and DIRECT were defined as "c:" and "\dir\" respectively. This results in

```
#include FILENAME "c:" "\dir\" "file.dat"
```

and with concatenation of the strings, this becomes

```
#include FILENAME "c:\dir\file.dat"
```

String concatenation is not restricted to the preprocessor, and it can be useful for printing large strings with printf as follows:

```
printf("variable 1: %d, variable 2: %d,"
       "variable 3: %d\n",i,j,k);
```

This is better than using the backslash method to continue long strings as this approach requires continuation lines to begin in column 1 as follows:

```
printf("variable 1: %d, variable 2: %d,\
variable 3: %d\n",i,j,k);
```

The Standard guarantees at least eight nesting levels for #include files.

If #includes are nested, subordinate headers are searched for relative to the location of the topmost #include unless an explicit location is specified in the nested #include.

The #define Directive

A new preprocessor-only "string-ize" operator # has been added to allow strings to be created during macro expansions as follows:

```
#define PRINT(i) printf("variable " # i " = %d\n",i)
```


INTRODUCING
DATALIGHT C

The Datalight C Compiler for MSDOS is a full C with all K&R constructs, including bitfields, plus the version 7 extensions. Other features of the compiler are:

$60

- Produces object files (.obj) so just the MSDOS linker is required.
- Floating point performed with 8087 or automatic software floating.
- Over 100 compact library functions with source.
- Compatible with the Lattice C compiler.
- Runs on IBM-PC, and compatibles, running MSDOS 2.0 or later.
- Complete, easy-to-read users' manual with index.
- Highly optimized code for production quality programs.

Datalight
11557 8th Ave. N.E.
Seattle, Washington 98125
(206) 367-1803

Outside USA add $10 shipping. Washington State residents add 7.9% sales tax. VISA and MasterCard accepted.

IBM-PC, a trademark of IBM; MSDOS, a trademark of Microsoft Corp.; Lattice C, a trademark of Lattice Corp.

Circle Number 7 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If the statement PRINT(count); is encountered, it is expanded as follows:

```
printf("variable " "count" " = %d\n",count)
```

and with string concatenation this becomes

```
printf("variable count = %d\n",count)
```

and, Hey Presto! We have dynamically created a string. Many compilers (including those based on UNIX) allow(ed) macros to substitute within strings; however, this practice is not permitted by the Standard, so the method demonstrated is necessary to achieve the same purpose.

Consider the following macro definition:

```
#define P(c) printf("char " # c " has the value %d\n",c)
```

If the argument c were the " character, a backslash would automatically be prepended to it before it were made into a string. The expansion would become

```
printf("char " "\"" " has the value %d\n",c)
```

and

```
printf("char \"
```

has the value

```
%d\n",c)
```

If two definitions for the same macro are encountered in the same scope, the second will be flagged if it is different from the first. Depending on how directives are processed, embedded white space may be significant. In other words,

```
#define ABC 10
```

and

```
#define ABC     10
```

may not be considered identical definitions. To avoid this problem, always use one space as the separator as this is guaranteed to be preserved. This concept is known as benign redefinition. This capability is particularly useful

new! version 3.0 WIZARD C

The following SIEVE benchmark was run without register variable declarations on an IBM/PC with 640K memory and an 8087.

	Exec Time	Code Size	EXE Size
Wizard C 3.0	**: 6.8**	**130**	**7,766**
Microsoft	:11.5	186	7,018
Lattice	:11.8	164	20,068

The new **Wizard C version 3.0** sets new records for all out speed! It leaves other C compilers in the dust! When your project depends on that last ounce of speed, choose Wizard.

Fast executable code, with multiple levels of optimization.

Six memory Models, supporting up to 1 megabytye of code and data, plus mixed model programming.

Effective error diagnosis, including multiple source file cross-checking of function calls.

A comprehensive runtime library, including fully portable C functions, MSDOS and 8086 functions.

8086, 8087, 80186 and 80286 hardware support.

Full Library Source Code included with package.

ROM based application support, including a ROM development package available to create Intel Hex files.

Fully ANSI compatible language features.

"...The compiler's performance makes it very useful in serious software development."

PC Tech Journal
January, 1986

" Wizard's got the highest marks for support."

"The Wizard Compiler had excellent diagnostics; it would be easier writing portable code with it than with any other compiler we tested."

Dr. Dobb's Journal
August, 1985

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language
February, 1985

(617) 641-2379 Only $450.

WIZARD SYSTEMS SOFTWARE, INC.

11 Willow Court, Arlington, MA 02174

MasterCard VISA American Express

Circle 49 on the reader service card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

in locating different definitions of NULL in separate headers.

```
#define NULL 0          /* in header 1 */
#define NULL 0L         /* in header 2 */
```

This situation can cause plenty of grief in using the Intel large model where the NULL pointer must be 0L, not just 0.

As a macro name is an identifier, it can have at least 31 significant characters in its name.

At least 1024 macro identifiers can be simultaneously defined in one source file (although no maximum character limit is set for the sum of their definitions).

At least 31 arguments may be used with a macro definition and subsequent invocation.

The Standard does not provide for "macro-stacking" or "macro-hiding." In such a case, a new macro definition does not replace an old one; it merely hides it until a subsequent #undef. This is not supported.

The #pragma Directive

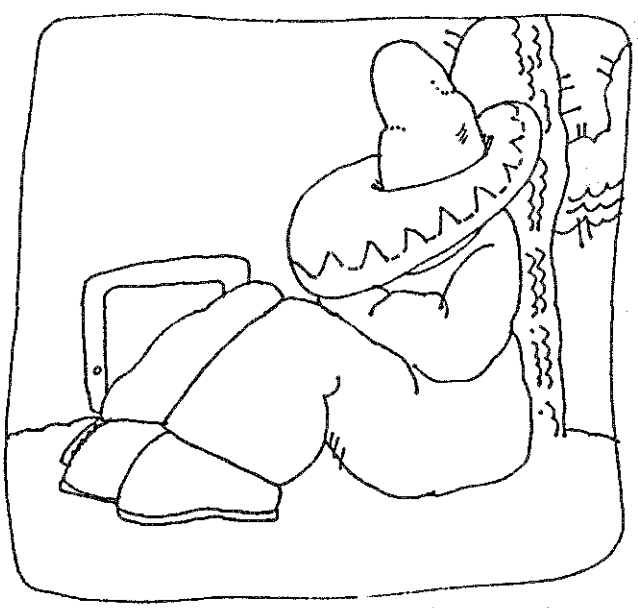
This implementation-defined directive has the form

```
#pragma ...
```

and can have any type of operand(s). If an implementation does not understand a #pragma, it ignores it without error. A pragma could be used to declare compilation listing arguments such as page width and size and other default compilation switches. It could also be used to change the level of checking for different blocks of code within the same source code file.

The null Directive

The naked directive # is still supported.



José takes a break from the runtime environment.

Predefined Macros

Several predefined macros exist that may prove useful. They are

__FILE__	=	current source file name
__LINE__	=	current source line number
__DATE__	=	translation date of the source
__TIME__	=	translation time of the source

All four macros have values in the form of string literals. The date format is "mmm__dd__yyyy," and the time format is "hh:mm:ss" (as a 24-hour clock), where the underscores in the date are really single spaces.□

Rex Jaeschke is an independent computer consultant, writer, and lecturer. He is also a member of the X3J11 Committee and co-founder and Editor of the C Journal.

If you would like to comment on or question any aspect of the Proposed ANSI C Standard, write to:

Rex Jaeschke
2051 Swans Neck Way
Reston, VA 22091

Rex will try to address your concerns in future columns as space permits.

C CODE FOR THE PC

source code, of course

Concurrent C	$45
Coder's Prolog in C	$45
LEX	$25
YACC & PREP	$25
Small-C compiler for 8088	$20
tiny-c interpreter & shell	$20
Xlisp 1.5a & tiny-Prolog	$20
C Tools	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Free shipping on prepaid orders No credit cards

Circle Number **30** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Best Debuggers. Period.

DSD86, The PC-DOS Debugger 69.95
DSD87, The PC-DOS Debugger with 8087 Support . 99.95
DSD80, The CP/M Debugger 125.00

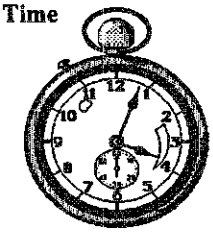
SoftAdvances

P.O. Box 49473 • Austin, Texas 78765 • (512) 478-4763

1-800-232-8088

Circle Number 33 on the Reader Service Card

Along With Your Computer, Your Time is the Most Important Thing You Own. . .So Why Waste It?



Quilt Programmer Productivity Tools will help you manage your software projects and get control of your time!

The Software Revision Management System (SRMS)™ stores all versions of source code/documentation in a single ASCII file. Allows retrieval of any version of source and application of changes while recording when, why and where changes were made with no duplication of common code. Other features include: full DOS pathname, directory, and environment variable support to help manage library maintenance; complete typeset users manual with tutorial; extensive error recovery procedures, support for most programming languages and much more!
SRMS (Version 2.0) $125.00

QMAKE™ is not just another make utility for MS-DOS. Sure it will accept a control file specifying dependancies of your software system, and rebuild your system by compiling only those routines that need to be compiled. It will analyze all dependancies before rebuilding any part of your system, and do it in a way that is extremely fast. Support for macros, multiple entry points, command line parameters, and most languages is also included. QMAKE differs from other DOS make utilites in that it completely integrates with the Software Revision Management System to give you a powerful combination of productivity tools.

QMAKE (Version 1.0).....$ 99.00
SRMS + QMAKE Combination.....$199.95

QUILT COMPUTING
7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

Circle 46 on the reader service card

WINDOWS GIVES YOU — LESS . . . . . *FOR*. . . . . MORE!

It's true. **WINDOWS** gives you less code and shorter development cycles for more sophisticated, eye-grabbing displays. Suddenly, pop-up windows, pull-down menus, and true context sensitive on-line help functions are a snap. You can drag windows around the screen and display different files in any number of windows. There's no snow and no delay.

WINDOWS is written entirely in assembly language so it's ultra-fast and VERY small: less than 2k (3k for large model). It runs on the IBM PC/XT/AT and compatibles. Being one object module, it's VERY EASY to learn and use.

The chapter on asthetics helps you pick the best color combinations for YOUR APPLICATIONS (hint: go for high contrast and avoid chromostereopsis). There's tricks for using placement, shape and color to maximize the communication quality of your programs.

WINGS LET'S YOU *FLY* THROUGH WINDOWS

For the ultimate in development environments, add ***WINGS*** to your **WINDOWS.** Our Window Generation System lets you create and modify windows in only seconds. A single keystroke then writes commented source code for your custom windows. With ***WINGS*** it's incredibly easy to experiment with different screen layouts and color schemes. Bottom line: your applications look much more impressive because you spend time optimizing the display, not writing code and recompiling over and over again.

WINDOWS is easy on your pocketbook — $99.95. ***WINGS*** is even easier — $49.95.

C WINDOWS provides ALL object modules for all memory models of Computer Innovations, Desmet, Lattice, and Microsoft C on the same disk. Support for Aztec and Mark Williams C is coming soon.

BASIC WINDOWS supports Digital Research CB-86 under PC/MSDOS, Microsoft Compiled Basic, Business Basic, and Quick Basic and the IBM Basic Compiler and Business Basic Compiler.

WE'LL INCLUDE FULLY DOCUMENTED SOURCE CODE FOR THE WHOLE **WINDOWS** package — **FREE**

Find out why our windows are in use at AT&T, the Canadian government, Digital Research, Heritage Bank, Lotus, Owens-Corning, Peat Marwick & Mitchell, Security Pacific, Scott Paper, Shell Oil, the U.S. Navy and many major universities . . . **Order today.**

CALL (213) 322-2853

SYSCOM SOFTWARE • 320 W. OAK STREET, EL SEGUNDO, CA 90245

Circle Number 34 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MICROSOFT 3.00 AND LATTICE 3.00

A Review of Two New Compiler Versions, and a Keyboard Input Function to Use with Both of Them

by William Hunt

Microsoft 3.0

Microsoft waited a long time before releasing an upgrade for version 2.03 of Microsoft C. Version 3.0 is an entirely new product and a big advance. Major changes relative to version 2.03 are

Compatibility: You will have to recompile your C source files and modify your assembler source files a bit. The manual describes differences pretty well.

Documentation: Microsoft C comes with two thick manuals whose typesetting and printing is first-rate. The *Library Reference Manual* is an unqualified success—good half- to one-page function descriptions in alphabetical order, good sections on topics such as text/binary mode I/O and descriptions of Microsoft C header files.

The *Users' Guide* covers use of the compiler, linker, and librarian programs thoroughly. Its 37 pages of installation and setup information and 41 pages of information on using the compiler is a bit overwhelming. The assembler interface section lacks examples, and information on the large model is dead wrong.

Reference Card: A short reference card lists the compiler's 38 options in three pages and is a lifesaver in everyday use.

ANSI C Features: Microsoft 3.0 supports most ANSI C features. (It does not support ellipsis marks to indicate optional arguments or dummy names for arguments in prototype definitions.)

New Library Functions: Version 3.0 nearly duplicates the function library of the Microsoft XENIX C compiler. A few functions for specific MS DOS services have been added. Neither Microsoft nor Lattice includes a full set of MS DOS specific library functions.

New LINK Program: Microsoft C includes Version 3.0 of the LINK linker, which supports large model overlays. It is also case sensitive, so you will want to upgrade to Version 3.0 of the Macro Assembler, which preserves case distinctions.

DOS Support: Microsoft 3.0 requires DOS 2.0 or a later version. Compiled programs won't run under DOS 1.1. This allows .EXE files to be a bit smaller. Library functions support DOS 3.1 file sharing and record locking features.

Compile Driver: The standard MSC command compiles just one file. An alternate command, CL, compiles and links a list of files and even recognizes wild cards. The CL command is not well documented in the manual.

Compiler Size: Version 3.0 is about 200 Kbytes bigger than Version 2.03 or Lattice C. A hard disk is a practical requirement.

Compile Speed: Version 3.0 is below average on compile speed. Compiling a series of files gets tedious.

.EXE File Size: Version 3.0 produces smaller .EXE files than Version 2.03 in most cases. Fewer library functions are included, and static arrays don't require .EXE file space in the small model. In the large model, arrays still add to .EXE file size. Lattice is ahead here with the new iAPX86 Version.

Math and Floating Point: Version 3.0 has a full set of math functions (exp, log, sin, . . .) as do current versions of Lattice C.Version 3.0 performs software float and 8087 float operations quite fast; Microsoft at the head of the field for now.

Register Variables: Version 3.0 makes effective use of register variable declarations, especially in the small memory model.

Language Reference Manual: A *Language Reference Manual* defines the C language as implemented by the Microsoft compiler. It is a more up-to-date and comprehensive reference than K&R, but it does not highlight its new features (such as function prototypes) very well.

Memory Models: Microsoft 3.0 supports the small (64 Kbyte code/64 Kbyte data), large (1 Mbyte code/1 Mbyte data) and program (1 Mbyte code/64 Kbyte data) memory models. Version 3.0 also allows a mixed memory model; you can mix 16- and 32-bit pointers. This option needs more documentation and library support to make it useful.

No Startup Source Code: Source code for the startup sequence is not provided. This makes the compiler far less attractive for writing systems software such as MS DOS device drivers, pop-up or memory resident programs, and ROM code. Library source code is not available either. The mixed memory model features just discussed would also be more useful with startup and library source code.

If you are still using Version 2.03 of Microsoft C, you should certainly upgrade. As the list of changes above shows, the Microsoft 3.0 compiler is a big improvement.

Lattice iAPX86 Version 3.000

Version 3.00d of Lattice C also incorporates ANSI C improvements. The list below covers major changes relative to Version 2.15e.

Function Argument Type Checking: Lattice Version 3.00 uses the function prototypes introduced by the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ANSI Standard to check the number and type of arguments in function calls.

Other ANSI C Features: The void data type, enumerated types and structures in assignments, function arguments, and function return values are all supported. The unsigned keyword can be used with char and long to create unsigned char and unsigned long data types. Nested comments are not allowed.

Statics Don't Take .EXE File Space: Uninitialized data declared outside a function or with the static keyword doesn't require space in the .EXE file. Programs with large global or static arrays will now produce much smaller .EXE files.

Faster Float: Execution of floating point arithmetic is faster using an 8087 and using software algorithms. A new compiler option to produce in-line 8087 code was still being tested when I reviewed the compiler.

DOS Specific Library Functions: The library has some new PC DOS specific library functions; functions for scanning a DOS directory are the most useful. Library function documentation lists numbers of DOS service calls used, allowing you to check the *PC DOS Technical Reference Manual* for more information.

More Documentation: At present, Version 3.00 comes with the 2.15 manual and a supplement describing changes in Version 3.00. Library functions changed in Version 3.00 are well documented in the supplement with all functions now in alphabetical order. An excellent index lists all functions and other external names with a one line description and classifies them as ANSI Standard, UNIX compatible, MS DOS specific, or unique to Lattice. A completely new manual is being prepared.

Compact Libraries: A new packing scheme makes the Version 3.00 libraries slightly smaller than the Version 2.15e libraries even though many new functions have been added to the library.

Signed Characters: Characters are now treated as signed numbers. You can use a compiler flag to force chars to be treated as unsigned values if you like.

Version 2.xx Compatibility: Compiler flags allow compatibility with earlier versions in unsigned char values, nested comment support, and multiple character constants ('abcd'). Source files that use buffered I/O (getchar/putchar for example) must be recompiled; other .OBJ files compiled with Version 2.15 may be compatible with Version 3.00.

UNIX Compatibility: Many library functions that mimic UNIX specific functions have been added to the library.

PC DOS 3.1 Compatibility: Library functions now support the file sharing and record locking features of DOS 3.1. Lattice provides two flavors of record locking to match various UNIX facilities. The file descriptors returned by open and creat functions are now PC DOS file handles and can be used directly in calls to DOS.

No DOS 1.1 Support: Library I/O functions now require Version 2.00 or later of MS DOS. Dropping DOS 1.1 support allows .EXE files to be 5 – 10 Kbytes smaller than for Version 2.15e.

Very Large Source Files: Version 3.00 distribution disks include a second set of compiler files (LC1L.EXE and LC2L.EXE) for handling very large source files (10,000 – 20,000 lines).

Version 3.00 has some significant improvements over previous versions of Lattice C, but it retains the same overall character. It has plenty of options and features but is one of the simplest C compilers to use. The compiler is fairly compact (considering its four sets of libraries) and compiles at a reasonable speed. If you like Lattice C, you will like Version 3.00 more. Upgrades from Version 2.15 are $45.00 and include new documentation.

Program sizes are much smaller for Version 3.00. For the filecop6.exe file this reflects the improvement in library overhead. In the lsort program, the global array of 8000 pointers produces an additional difference. Execution time for software floating point is also improved.

The table below compares some benchmark results for Lattice Versions 2.15e and 3.00d and Microsoft 3.00.

	Lattice 2.15e	Lattice 3.00d	Microsoft 3.00
Compile Times (secs.)			
59 lines	15.0	23.0	24.7
781 lines	68.6	72.5	98.5
Program Sizes (K bytes—small/large model)			
LSORT	27K/46K	7K/8K	6K/40K
FILECOP6	11K/14K	5K/5K	4K/4K
Execution Times (secs.—small/large model)			
quicksort	20.0/26.5	20.0/26.5	16.7/26.0
file I/O (getchar/putchar)	14.0	17.0	10.0
float s/w	54.6	39.2	31.5
float 8087	2.8	1.0	1.0

Compile times for Lattice C are noticeably shorter than for Microsoft C, especially when a series of files is compiled. Program sizes are slightly smaller for Microsoft C except for large model programs with global or static data arrays.

Microsoft 3.00 is significantly faster than either version of Lattice for buffered I/O. Neither compiler provides a really fast set of buffered I/O functions; if buffered I/O speed is crucial in your application, plan to implement your own functions. Performance using the read/write unbuffered I/O functions is the same for Lattice and Microsoft.

In floating point speed, Lattice 3.00 matches the speed of Microsoft 3.00. The Lattice 3.0d result does not use the in-line 8087 option, which was not fully tested when the benchmarks were run.

Microsoft C and Lattice 3.00 are functionally equivalent. Microsoft C is ahead in some areas: use of register variables in the small memory model, and exten-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

sive documentation. Lattice is ahead on compile speed, PC DOS support, and large model program size. At present source code for the startup function (C.ASM for Lattice) and the library functions is not available for Microsoft C. This makes Microsoft C a poor choice for writing memory resident software, device drivers, or other low-level software.

If you are reasonably satisfied with your current vendor, you should probably stick with that compiler. The differences between these compilers are not large enough to warrant switching compilers.

This review of Microsoft and Lattice C 3.0 focuses on improvements relative to previous versions of the compilers. The purpose is to show what you get if you upgrade. The benchmarks shown here are not extensive enough to show the differences between the compilers.

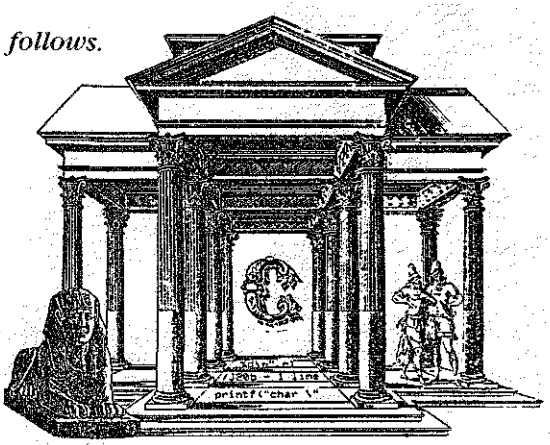
A Keyboard Input Function

The Lattice C and Microsoft C 3.0 libraries provide the getch function for getting a single character input from the keyboard without echoing it and without waiting for the return key to be pressed. This function gives us access to all of the PC's keys, even those, such as the arrow keys, that do not correspond to an ASCII character. However, DOS represents such non-ASCII keys with two characters—a NUL character followed by a modified scan code. This is awkward to handle in our programs, so the getk function in getk.c builds on getch to make a function that returns with one keystroke for each call. Non-ASCII keys are returned as values of 0 x 100 or 256 decimal plus the modified scan code.

The keys.h file below defines values returned for cursor control and function keys. To get the value returned for function key Fn add (n–1) to the corresponding value for F1. For example, pressing F5 returns 309 (F1 value + 4), and Shift F5 returns 344 (Shift-F1 + 4).□

Bill Hunt writes the Lattice C Users Group *newsletter, where this material first appeared, and is also the author of the* C Toolbox, *published by Addison-Wesley and currently on B. Dalton's Best Seller list.*

Code follows.



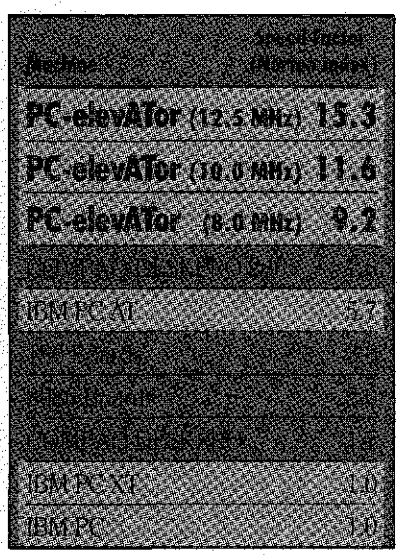
```
/* getk.c - get one keystroke */
#include "stdio.h"

int getk()                              /* get next keystroke */
 {                                      /* (may be two chars) */
    int c ;

    c = getch() ;
    if( c == 0 )                        /* is it non-ASCII ? */
      c = 0x100 + getch();              /*    y - get scan code */
    return( c ) ;
 }

/* keys.h - values returned by getk for non-ASCII keys */
        /* definitions of cursor keys */
#define UP_ARROW  (256 + 72)
#define DOWN_ARROW (256 + 80)
#define LEFT_ARROW (256 + 75)
#define RIGHT_ARROW (256 + 77)
        /* F1 - normal,shifted,control,alt */
#define F1_KEY  (256 + 59)
#define F10_KEY (256 + 68)              /* F10 value = F1 + 9 */
#define SF1_KEY (256 + 84)
#define CF1_KEY (256 + 94)
#define AF1_KEY (256 + 104)
```

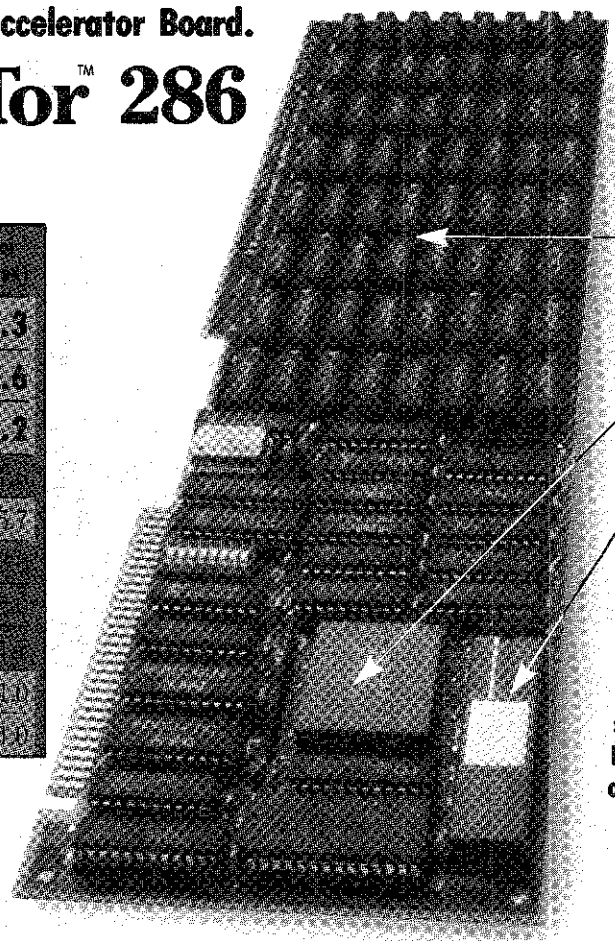

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

WE START WHERE OTHERS LEAVE OFF.

The World's *FASTEST* Accelerator Board.

PC-elevATor™ 286





"If my primary tasks centered around calculation-bound programs, such as 1-2-3 or SuperCalc, PC-elevATor would be a clear winner in my mind."

BUSINESS SOFTWARE
January 1986

A powerful family of 80286 board products.
PC-elevATor! The ultimate plug-in processor. Now with three versions of the 80286 chip:

FAST — 8.0 MHz
FASTER — 10.0 MHz
FASTEST — 12.5 MHz

PC-elevATor. Outperforms all turbo cards. And only with PC-elevATor can you upgrade to a faster board as your processing needs grow.

With up to 2 megabytes of on-board extended memory.
High-speed 16-bit RAM. AboveBoard™ (EMS) support. RAM disk. Disk cache. Print spooler. Optional 80287 coprocessor. One board is all you need. PC-elevATor.

For PC compatibility, and more.
Boosts your PC, XT, AT, COMPAQ or compatible to new speed limits. Optimizes the 3270 PC, freeing a full 640K for PC applications while existing PC memory runs 3270 Control Program. Just plug and run. No chips to remove. No cables to connect.

"PC-elevATor is a formidable information processor."
PC WEEK, 9/24/85

"It pulverized the sorting, calculating and search-and-replace tests."
BUSINESS COMPUTER SYSTEMS, January 1986

"A very speedy and promising product. Screen handling is superb."
PC MAGAZINE, 11/12/85

Technical Features
- 8, 10 or 12.5 MHz 80286
- 5 or 8 MHz 80287 (optional)
- 1 or 2 Megabyte RAM (No Wait-states)
- Operates in tandem with PC
- Supports EMS/Above Board
- Includes RAM Disk, Disk Cache, Print Spool

Call today to find out how the PC-elevATor family can meet your current needs and support your future growth.

APPLIED REASONING

Applied Reasoning Corporation
765 Concord Ave., Cambridge, MA 02138
(617) 492-0700 TELEX: 6714194 elevator

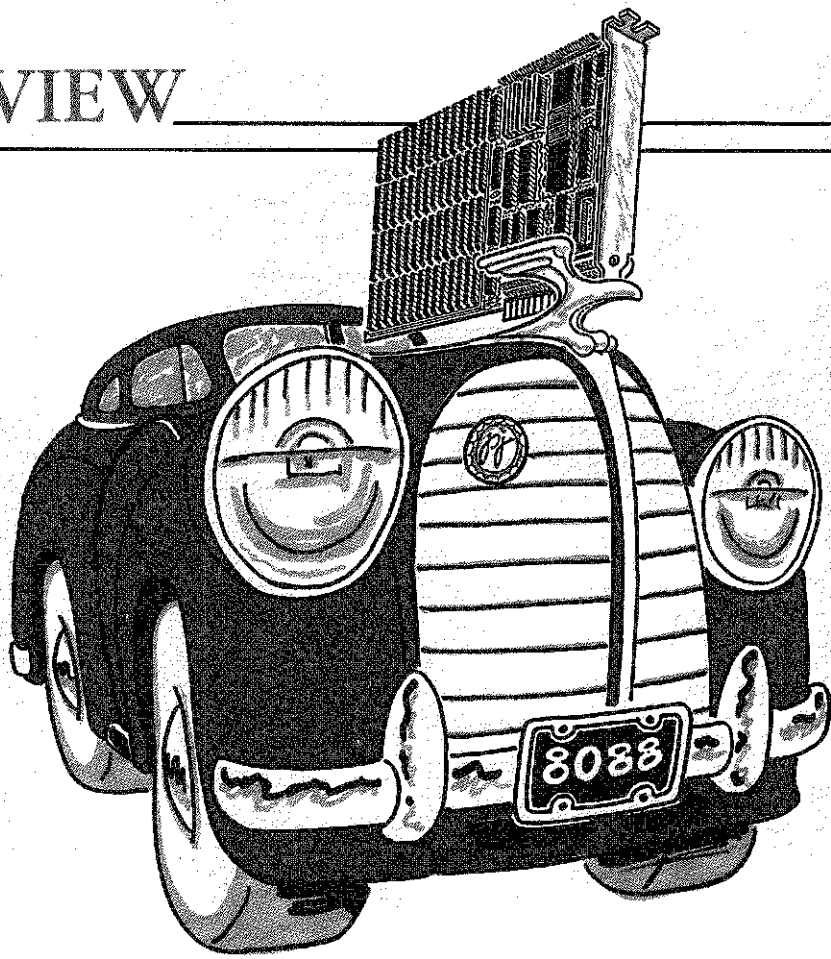
PC-elevATor is a trademark of Applied Reasoning Corporation. COMPAQ DESKPRO and COMPAQ DESKPRO 286 are registered trademarks of COMPAQ Computer Corporation. IBM PC, XT and AT are registered trademarks of International Business Machines Corporation.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

O! Call back yesterday, bid time return

—King Richard II

PJ TRIES OUT SOME TIME-SAVING 80286 COPROCESSOR BOARDS



Thanks to Steve Baker for his work on benchmarks.

A number of Intel-family coprocessor boards have appeared lately, and the most recent group features the 80286. These boards don't actually turn back the clock, but it may seem that way. They just might help you shape up your software development in record time.

A General Overview of 80286 Coprocessor Operation

The 80286 instruction set is a superset of the 8088 set, so code written for the 8088 can be executed by the 80286. However, the 8088 and 80286 don't mate without some difficulties.

The Buses

The 8088 and the 80286 have different bus sizes. Accelerator board companies have dealt with this incompatibility in two basic ways.

The I/O Approach

One approach is to build nearly a complete 80286 computer on a PC board and to interface the 80286 bus with the 8088 bus through bus I/O. The 80286 typically has its own memory (1 to 4 Megs of fast, no-wait-state RAM), with part of it *dual-ported* so that it can be read by both the 8088 on the motherboard and the 80286.

When the board is installed, the computer still boots normally with the 8088. A DOS-level software command can then transfer control to the accelerator board. (For our review boards, the commands were *PFAST, TURBO*, and *UP*.) DOS is booted onto the coprocessor board, and the 8088 is left running a program that turns it into an I/O server. The user sees a normal DOS prompt and has normal access to the disk drives except that it is now a 286 executing DOS instead of an 8088. The user can now run DOS applications normally.

Another command (*PSLOW, UNTURBO, DOWN*) will take you back to the DOS environment on the motherboard.

Screen Handling

The 8088 attends to the 286 board's interrupt, and, typically, polls the dual-ported memory for screen writes during the timer interrupt routine. When a program that is running on the 286 attempts to do a direct memory write, it actually only writes into this dual-ported memory area, and the 8088 picks up the data and moves it to video memory.

There are pitfalls for software running on such a board. Hardware and software interrupts must be redirected to the board, DOS and BIOS calls must be emulated, "badly-behaved" software anticipated.

The 8088 Socket Approach

The other approach is to remove the 8088 and plug a board directly into the vacant 8088 slot.

This allows more direct access to the PC motherboard hardware and I/O. Here, typically, the 80286 bus accesses a small cache memory. 8088 code is fetched into this cache before it is executed so that the code can be executed without wait states and a word at a time instead of in byte chunks.

This approach appears to be slower than the I/O board approach, however, because of the continual cache fetches and because wait states must be inserted during motherboard memory access. It, too, has its difficulties—for starters, with software dependent timing loops, copy protection, slow hardware, and with software that continually jumps around in the code making cache fetches inefficient.

Software Development

There is one application, though, that is a natural for both approaches—software development. Compilers are

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

generally well-behaved, processor-intensive programs, and time is important when writing software.

Many of the accelerator boards also retain the 8088 so that work can be compiled on the 286, then quickly transferred to a vanilla 8088 environment for test.

So what can a 286 coprocessor board do? To find out, *PJ* compares eight software development environments:

The standard 8088 PC
The standard PC with a NEC V20 CPU
The 8086 AT&T 6300
The 8086 AT&T 6300 with a NEC V30 CPU
The PC AT
Three 286 accelerator boards running on a PC:
Orchid's PCturbo 286e, the Phoenix Pfaster 286, and Applied Reasoning's PC-elevATor

The compilers used were Aztec C 3.20B and Turbo Pascal 3.0.

Sieve

First, 100 iterations of the standard Sieve of Erasthenes in C (compiled with Aztec 3.20B with no register variables).

The time is in seconds.

PC	PC/NEC	6300	6300/NEC	AT	Orchid	Phoenix	A.R.
105	103	50	43	36	21	20	20

The sieve was timed by a program that made a call to the on-board clock at the beginning and the end and returned a time accurate to the nearest second.

Version 3.1 Norton Index

The Norton Index (included in the Norton Utilities as the program *SI*, or *system information*) is a measure of processor speed and memory access that has become popular with accelerator board manufacturers, possibly because it returns pretty attractive figures. An 8088 PC scores 1.0 on the index. A 3.5 means that in terms of pure processor speed and memory access time, a machine scores 3.5 times as fast as a PC.

(average of 20 passes each)

PC	PC/NEC	6300	6300NEC	AT	Orchid	Phoenix	A.R.
1.0	1.75	1.7	3.5	5.0	8.35	8.24	8.345

The Version 3.1 Norton Index appears to return lower scores than the Version 3.0 index, which might explain why accelerator board manufacturers have previously reported scores above 9.0.

There's more to the coprocessor boards than CPU speed and memory access though—there's the speed of DOS and BIOS emulation.

MAKING THE 286e COMPATIBLE

by Robert Hurt, Project Manager for the Orchid PCturbo 286e

Normal computational features of the 8088 are easily handled by the 80286 because the 80286 is upward-compatible. The only exception to this is programs that depend absolutely on exact execution times of particular instructions. These programs will either run too fast (as in the case of some games), or will not run at all (as in the case of old-style [1983 vintage] copy protection schemes). Using the 'unturbo' command, you can switch back to the 8088 to run any of the troublesome programs you might have.

There are several mechanisms built into the Turbo to allow it to run most IBM PC programs. Almost all of the mechanisms have to do with the handling of I/O: Video I/O, Disk I/O, and Port I/O.

Video I/O

When a program on the Turbo writes to video memory, it writes to actual RAM on the Turbo. This is not video memory, so circuits on the Turbo set a flag when a program writes to an area of video memory. Flags are available for 16 areas of memory. The program running on the PC can look at these flags. When it sees a flag set, it opens a window into that area of Turbo memory and simply moves the data from there to the corresponding area of real video memory. Since it is faster to move several kilobytes in a block rather than a character at a time, the Turbo has very fast screen I/O.

However, this speed has a price in compatibility with the enhanced graphics adapter (EGA).

The EGA

The EGA card is nice because it allows better resolution and more colors than the CGA (color graphics adapter). However, it does not use a straightforward method of accepting pixel information from a running program because it requires more display memory than is standardly available in a PC. Its designers chose to use a combination of I/O commands and data transfers into video memory in order to paint the display. These MUST be synchronized in order to display correctly. That is where problems come in.

The program running on the PC does not know about the requirement for synchronicity. The PC checks for video flags only every so often. It could be that several I/O instructions occurred synchronously on the Turbo with several interspersed writes to video memory. Therefore, it transfers the video memory asynchronously with doing the I/O instructions. As a result, the proper sequence of I/O instructions and video writes for an EGA board cannot occur.

If the synchronicity were forced anyway (as it could be with another design), screen updates would be drastically slowed, and the EGA would be useless. So the Turbo works with standard EGA boards only when they are in CGA mode.

The Turbo works fine with CGA, Monochrome, Hercules, and Tecmar Graphics Master. These boards do not require a synchronized interspersing of I/O commands and writes to video memory, and they use a reasonably small video memory space (160 Kilobytes or less).

Continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The PCTurbo 286 does provide a way around the compatibility problem with EGA-style graphics adapters. That is the TurboBus, a connector that makes major 80286 bus signals available through 80 pins to a user-supplied ribbon cable at the top of the board. The connector allows attachment to custom cards so that they can communicate at higher speed than allowed by the PC's bus. It is feasible that a custom EGA card could use either the 8088 processor via the PC bus or the 80286 processor via the TurboBus. This would not only provide compatibility with EGA, but faster drawing as well.

Disk I/O

Disk I/O is handled simply by the Turbo. Since it cannot be known in advance whether the Turbo will be plugged into an AT or XT or PC, there is no way to tell what kind of disk or disk-handling DOS/BIOS will be on the host computer. Therefore, when a DOS interrupt 21 or BIOS interrupt 13 for disk I/O comes from an application program, the Turbo interrupts the 8088 so that the PC's DOS and/or BIOS can handle it. The Turbo is compatible with almost any host computer disk configuration.

Port I/O

The final form of I/O handling is the trapping of I/O instructions. Obviously, the Turbo's 80286 is not directly connected to the host computer's I/O ports. The Turbo handles application program accesses to the ports by trapping them and notifying the program on the host computer that I/O is to be done. The host CPU does the actual I/O or emulates it. The benefit of this method of handling I/O is that the application program seems to be running directly on a PC. That is, you get 100% functional compatibility. However, the I/O can be slower than it is on the real PC because of the overhead in the trapping mechanism. Realistically, for example, maximum serial port communication speed is 1200 baud on one port. Higher baud rates may cause incoming data to be lost.

Using EMS

The Turbo can use EMS memory, and the user can buy an optional 1MB EMS RAM daughter card for the Turbo for $395. This is not normal RAM. It is the same 120nanosecond RAM that comes on the Turbo, and it is soldered in for reliability. The card also operates in extended mode so that the 80286 can address above 1MB when running in Protected mode. This allows UNIX and XENIX to be ported to the Turbo with relative ease, a task that Orchid has been asked about recently.

Productivity Software

The productivity software includes TurboDrive RAM Disk, TurboCache Disk Caching, and Print Spooling drivers.

TurboDrive and TurboCache use either or both the PC and Turbo memory (that is, both DOS memory and EMS/Extended memory) to hold disk data for throughput improvements. The Disk Cache can be automatically invoked in the host computer's DOS memory (half a megabyte or more) and EMS memory (up to 2 Megabytes on Orchid's Conquest multifunction card) simultaneously. Practically all the programs and data you use all day long will fit in the cache with just one access to disk for the initial reading. And the print spooler holds huge files for the printer while the Turbo goes on with other computing tasks.

Continued . . .

Compile to Hard Disk with Overlays

This test is a compilation of 500K of source code in 14 include files into a .COM file and two overlay files—a test of processor speed and DOS disk I/O emulation in a typical software development chore.

This test was run on a single PC and tested the PC in 8088 mode vs. the three accelerator boards. The state of the hard drive was an important factor that couldn't be duplicated on the other machines, so they weren't included.

The compile time is in minutes

PC	Orchid	Phoenix	A.R.
8:05	2:04	2:27.3	2:29.2

Compile to RAM Disk in 286 Protected Memory

Protected memory is the memory beyond DOS that the 286 accesses by generating the segment value from a table in a way designed to (1) protect memory in a multi-user environment, (2) make programmers cry. Very few programs have made use of protected-mode memory, and most software firms may just wait the situation out because the 386 reportedly uses a 32-bit offset. You can just keep the 386 segment register at 0 and write all your code to a huge linear address space. The next version of DOS reputedly will also handle protected memory problems for 286 machines.

RAM disks that can use protected memory are around, though. The Phoenix board was the only one that came to us with protected memory, so we put a 1 Meg RAM disk up there, moved the 500K of source code onto it, and compiled to the RAM disk.

PC	Orchid	Phoenix	A.R.
n/a	n/a	1:59	n/a

The next three benches were compilation of a 50K source file. No overlays involved.

50K Source File to Hard Disk

PC	6300	AT	Orchid	Phoenix	A.R.
41.5	21	17.5	10.2	11.5	11.2

50K Source File to Memory

PC	6300	AT	Orchid	Phoenix	A.R.
41	20	17	9	10	9.5

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

50K Source File to Floppy

PC	6300	AT	Orchid	Phoenix	A.R.
0:46	0:28	0:23	0:15.5	0:18	0:16.2

Write to screen with Bios call

As a test of screen handling, the next test writes 20,000 characters to the screen with a BIOS call.

PC	6300	AT	Orchid	Phoenix	A.R.
0:28	0:31	0:14	0:05	0:08	0:09

As a test of typical text editor work, the next two tests load and save a file from Personal Editor, IBM's workhorse. These benches are a combination of Hard Disk I/O, memory access, and processor work.

Load File, Personal Editor

	PC	Orchid	Phoenix	A.R.
hard disk:	0:55	0:12.8	0:18	0:16

Save File, Personal Editor

	PC	Orchid	Phoenix	A.R.
hard disk:	1:27	0:21	0:22	0:24.5

As a test of DOS disk I/O:

DOS File Copy, Hard Disk to Floppy

PC	6300	AT	Orchid	Phoenix	A.R.
0:14	0:16	0:13	0:13	0:17.5	0:15.5

DOS File Copy, Hard Disk to Hard Disk

PC	6300	AT	Orchid	Phoenix	A.R.
0:05.5	0:06	0:03	0:04	0:05	0:06.8

Using different hard disks introduces an unknown amount of uncertainty, but the times provide a rough guide to speed.

Parallel Processing and Other Enhancements

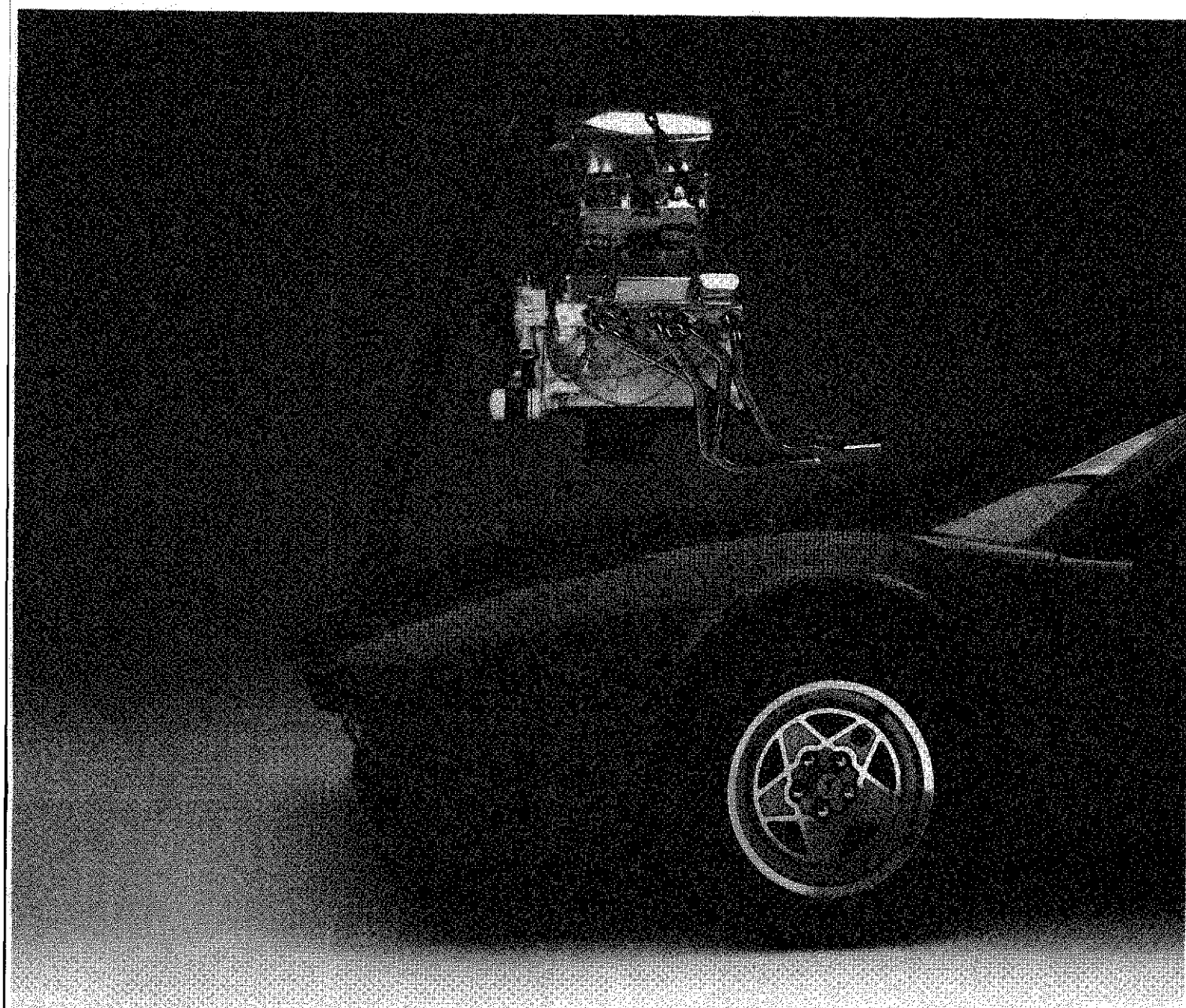
Orchid designed the Turbo so that it will work not only in a PC, AT, or clone, but also in concert with other PCTurbo 286e's in the same system. Since the Turbo uses a jumper selection to determine the I/O ports through which the PC commands it and a programmable 64K window address through which the PC can access the Turbo's dual-ported memory, you can put several in the PC at once. We have customers who are using parallel processing software to pass tasks and their results to and from multiple Turbos in an AT. Although multiple Turbos can be used in the same system, customers must write their own software to make use of them. Standard software shipped with the Turbo allows only one in the system.

Orchid is investigating a number of software enhancements for the Turbo. While it is already compatible with SNA (3270, etc), we are looking at networking and concurrent processing.

The PCTurbo 286e really shines because it is VERY compatible, allows you to switch back to 8088 mode when compatibility is a problem (which is seldom if ever for most users), and runs like a bat out of h--. If absolute compatibility with everything in the world were the issue, the user would have to stick to the gloomy old PC. □

Melvin Makefood, chef at Cafe' Boca Raton, is said to have developed the original graphic syntax.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



AT™ Pfantasies for your PC or XT.™

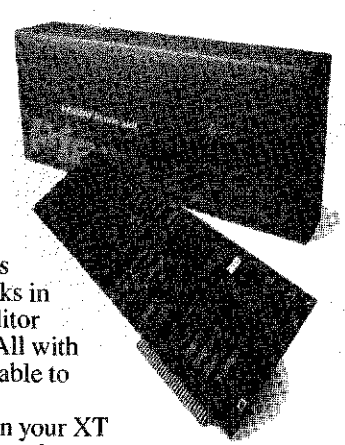
Want better speed and memory on your PC or XT without buying an AT?

You've got it!

Phoenix's new Pfaster™286 co-processor board turns your PC or XT into a high-speed engine 60 percent faster than an AT. Three times faster than an XT. It even supports PCs with third-party hard disks. But that's only the beginning.

You can handle spreadsheets and programs you never thought possible. Set up RAM disks in both 8088 and 80286 memory for linkage editor overlays or super-high-speed disk caching. All with Pfaster286's 1mb of standard RAM, expandable to 2mb, and dual-mode design.

You can develop 8086/186/286 software on your XT faster. Execute 95 percent of the application packages that run on the AT, excluding those that require fancy I/O capabilities your PC or XT hardware just isn't designed to handle. Queue multi-copy, multi-format print jobs for spooling. Or, switch to native 8088 mode to handle hardware-dependent programs and back again without rebooting. All with Pfaster286's compatible ROM software. And, Pfaster286 does the job unintrusively! No motherboard to exchange. No wires to solder. No chips to pull. Just plug it into a standard card slot, and type the magic word, "PFAST."



If you really didn't want an AT in the first place, just what it could do for you, call or write: Phoenix Computer Products Corp., 320 Norwood Park South, Norwood, MA 02062; (800) 344-7200. In Massachusetts, 617-762-5030.

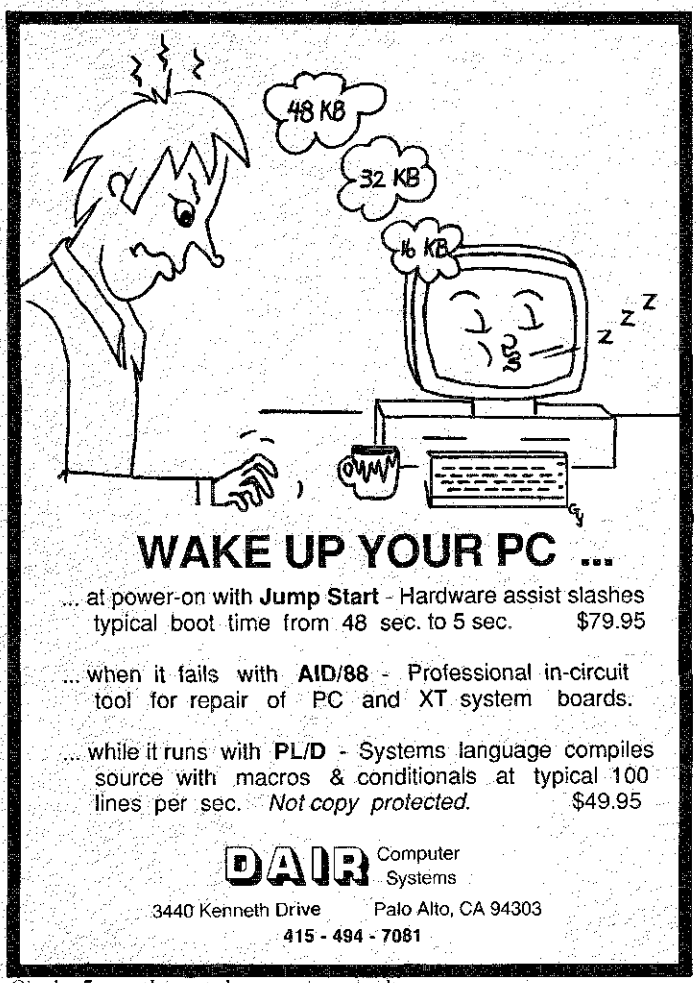
Programmers' Pfantasies™ by

Phoenix

XT and AT are trademarks of International Business Machines Corporation. Pfaster286 and Programmers' Pfantasies are trademarks of Phoenix Computer Products Corporation.
For the Ferrari aficionado: yes, we know this is a rear engine car. We are showing the addition of a second engine to symbolize how Pfaster can be added to your PC or XT to increase performance.

Circle 51 on the reader service card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


48 KB
32 KB
16 KB
z z z
WAKE UP YOUR PC ...
... at power-on with Jump Start - Hardware assist slashes typical boot time from 48 sec. to 5 sec. $79.95
... when it fails with AID/88 - Professional in-circuit tool for repair of PC and XT system boards.
... while it runs with PL/D - Systems language compiles source with macros & conditionals at typical 100 lines per sec. Not copy protected. $49.95
DAIR Computer Systems
3440 Kenneth Drive Palo Alto, CA 94303
415 - 494 - 7081
Circle 5 on the reader service card

Orchid's PCturbo 286e
Turbo and Unturbo
Orchid Technology
47790 Westinghouse Drive
Fremont, CA 94539
(415) 490-8586

$1195.00

8 MHz 80286 coprocessor board

Hardware

See the related article from Robert Hurt, Orchid project director.

Software

An installation program creates a subdirectory called *PCTURBO* on your boot disk, inserts the subdirectory into your DOS path and copies the needed files into the subdirectory, and appends your AUTOEXEC file or creates one.

The commands *Turbo* and *Unturbo* move you from host to board and back.

The board normally emulates an XT. The manual says that optional emulation of the AT allows the 286e to run UNIX, but that the standard board won't handle UNIX or other software that uses AT I/O hardware.

Also included are Disk Cache, Print Spool, and RAM Disk software.

The current version of the 286e does not work with networks. The 286e can be in a system that is on a network, as long as the board is not active. Orchid reports that they are currently testing networking software.

The Manual

The manual to the 286e is spiral bound, thorough, and well done.

Running the 286e

Installation of the 286e was uneventful. It was running 20 minutes after the box was opened. We ran it on an XT clone with 640K on the motherboard, a monochrome monitor, two floppy drives, and a 20 Meg hard disk.

After a day, the board started signaling parity errors on boot and also failed to boot a few times. Orchid quickly replaced it with another, which thus far has just run and run. Orchid pointed out that the first board was a Beta unit. The second board also came with Version 1.1 of the software, which included a couple of commands that were just stubbed out in Version 1.0.

The only software difficulty we have found was when running an application that handles a 32K baud I/O stream through a dedicated card serviced by IRQ3. When the stream became heavy, we began to lose data, something that didn't happen when running on the 8088.

Our board didn't have the daughter card, so we weren't able to test an EMS or Protected RAM disk.

The Phoenix Pfaster286
Pfast and Pslow
Phoenix Computer Products
320 Norwood Park South
Norwood, MA 02062
(617) 762-5030

$1495.00

8 MHz 80286 coprocessor board

Hardware

The Phoenix board with 2 Megs on board is a gleaming mother: rows of 256K DRAMs and an unmistakable black heat sink over the 286.

The Pfaster comes in three models: Pfaster 1 with 1 Meg (which cannot be expanded to 2 Meg), Pfaster 2 with 2 Meg and VDISK.SYS capability, and Pfaster 3 with 2 Meg, VDISK.SYS, and ViaNet LAN software.

The board features the AT extended memory support of ROM call INT 15, so applications can access up to 4 Megabytes. It supports 8 levels of priority interrupts and 4 DMA channels.

The 2 Meg board with 80287 option draws a maximum of 12.35 watts. You'll probably need a 135 watt power supply.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Pfaster's status, control, and data ports are memory mapped into the E000 segment of the board. This 64K segment is reserved for the board's BIOS ROM emulation and support routines. The 8088's interrupts are redirected into the board, and the board passes as many of the interrupts to the 8088 as possible. The 8088 returns the results.

The board reserves the 64K starting at b000 for the screen buffers. Data in this segment is offloaded by the 8088 during the timer interrupt. The normal update speed of 18.2 times per second can be varied at boot time. If an application program running on the board attempts to get the status of the vertical retrace bit, the value is returned as set every other time it is requested.

Int 16 hex, the keyboard service routine, is handled directly by the board's ROM, which traps out Ctrl-Alt-Del, giving you the option of returning to 8088 mode instead of warm booting the system.

Software

The software includes an INSTALL utility, diagnostic routines, and various system and batch files.

Booting

The program PFAST loads DOS onto the board and remains running on the 8088. A number of software switches can be set on the command line when starting PFAST—memory, drive and hardware interrupt handling options. Each interrupt can be handled by the original 8088 routines, or by a combination of the 286 and 8088.

One of the system files, GLDD.SYS, passes control of installed drivers on the 8088 side over to the 80286 side when the board is booted.

The PC ROM emulating routines are contained in a file, PFASTROM, which is loaded onto the board before the copy of DOS is loaded. At boot, all of the low memory interrupt vectors are redirected into the PFAST286 ROM emulation software.

Once on the 286 side, pressing Ctrl-Alt-Del opens a window on software switches which will again specify how each hardware interrupt will be handled.

The software also includes a RAM disk and a cache handler.

Running the Pfaster286

There were a couple of problems with installation, but nothing that an experienced PC user couldn't manage.

The software came with the new INSTALL utility that was supposed to create an autoexec file that would boot you directly into the 286, but boot proceeded to the 8088. The Phoenix software consists of 20 files, and it took a while to sort through them to ascertain the reason.

Finally, it became apparent that all the .SYS files and .BAT files needed to be brought out of the PFAST subdirectory created by INSTALL and onto the root directory. No major problem.

Once this was straightened out, the board ran without problems.

If BASIC's data limit is the question . . .

DataSpace

. . . is the answer

DataSpace gives MS* and GW-BASIC* users . . .

- the ability to define and access string, integer, single, and double precision data arrays
- 10 - one, two or three-dimensional arrays up to 64K each - 640K total
- array storage and retrieval from files in any disc directory
- full compatibility with interpreters or compilers

. . . all with absolutely no impact on BASIC's own data space (requires MS-DOS* 2.10 or higher)

The only answer around for $75 . . .

732 Sixth Avenue
Aurora, IL 60505
(312) 898-0651

* MS, GW-BASIC, and MS-DOS are trademarks of Microsoft Corporation

Circle **41** on the reader service card

The Manual

The Pfaster manual comes in a three-ring binder and includes an index. It is thorough and contains an assembler INCLUDE file with the definition of the status, control, and data ports (that are mapped into the E000 segment).

Applied Reasoning's PC-elevATor
Up and Down
Applied Reasoning
765 Concord Avenue
Cambridge, MA 02138
(617) 492-0700

$1195.00

8MHz 80286 coprocessor board with 1 Meg RAM
Software Version 1.46
Manual Draft Version 3.2

Hardware

Occupies one long slot.
Power: 12 watts.
80286 running at 8MHz with no wait states. Optional 80287 selectable at 4.8, 5.3, or 7.2MHz.
512K to 2 Meg of memory on board, 256K 150nanosecond DRAM, 128 or 256 cycle refresh.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Sort RM/COBOL Files

• • • **QISORTC** • • •

Finally, a sort utility for RM/COBOL.
Using efficient algorithms in assembly language making it the fastest sort on the MS-DOS market. Qisortc supports:

- RM/COBOL Version 1.5x & 2.xx files
  - Single / Dual File Indexed
  - Relative and Sequential
- ASCII, COMP-1, & COMP-3 fields
- 32 sort control fields
  - ascending / descending
- 32 record selection fields
  - EQ:NE:GT:GE:LT:LE field:constant
  - implied AND, OR may be specified
- 32 output fields or total record
- CALL from RM/COBOL, when provided subroutine is linked with runtime
- Use pathnames and return codes
- Keyboard or Parameter file
- Requires MS-DOS ver 2.xx or above
- Only $95.00 - COD, VISA, or M/C.

Quantumn Information Systems, Inc.
145 N.W. 85th St., Suite 103
Seattle, Washington 98117
(206) 789-2888 Telex 4941041

Circle Number **31** on the Reader Service Card

I/O addresses selectable from 31C-31F, 21C-21F, 11C-11F, or DC-DF.
Interrupt number: selectable from 2,3,4,5,6,7 or none. DMA channel not required.

Most board options are set with a DIP switch instead of jumpers, which is handy.

The PC-elevATor automatically positions its 128K dual-ported interface with the 8088 in an unused address segment between D0000 and EFFFF. The manual mentions that some devices hide their RAM or ROM usage of this area until they are running. To resolve any conflicts that might occur, use the UP.SYS file to force the interface window at a specific address.

To handle screen writes, the UP.SYS file provides three options: COPYMETHOD = FAST, COPYMETHOD = MEDIUM, and COPYMETHOD = SLOW. FAST most probably takes no account of vertical retrace, and SLOW probably does. The manual says that the MEDIUM method will cause "some" snow. It isn't specific about the methods employed in any of the three cases.

Software

The software is straightforward and simple to use. An INSTALL command works well. If you want to create a subdirectory for the board's software and put it on the DOS path, you've got to do it yourself, but that's simple.

The file UP.SYS contains ASCII instructions to the board boot program. The parameters specified in UP.SYS and their meanings are contained in Appendix A of the manual and allow you to tailor UP.SYS in a wide variety of ways.

Using UP.SYS instructions, you can place a DMA window in the dual-ported memory, halve the speed at which the 80286 runs, and insert a wait state in each memory cycle, among other things.

The software contains a number of files that enable compatibility with software packages. Ethernet and Novell networks are supported.

The PC-elevATor doesn't include RAM disk software, but it is compatible with the VDISK software included with PC DOS 3.00 or later.

The Manual

The Manual is well-written and clear.

Running the PC-elevATor

The PC-elevATor board was simple to install and run. It was up and running in minutes and performed flawlessly.

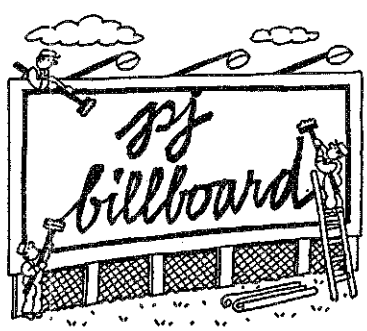
Conclusion

The Phoenix and Applied Reasoning boards showed a slight edge in processor and memory access speed over the Orchid board in the Sieve benchmark, but the difference (20 vs. 21 seconds) was within the range of the test's timing uncertainty.

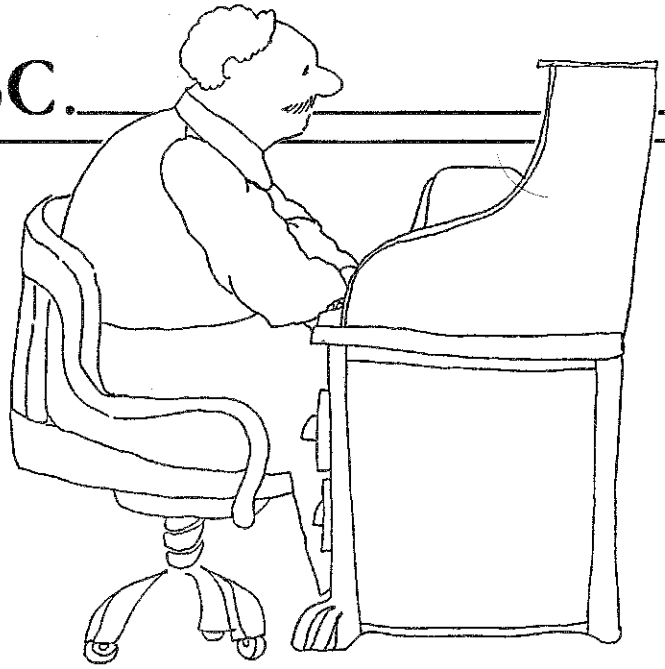
The Orchid board appears to be quicker at software development tasks that include disk I/O and screen writes, probably due to better emulation of DOS and BIOS calls. The Orchid board appears to have more supporting "glue" chips than the other two, including a VLSI gate array. But there are many variables involved in I/O, especially with these boards. We tested them in their default configurations, but since all three are reconfigurable, it is possible that their disk caches, screen updates, and other factors could be tweaked to produce different results.

All three boards performed exceptionally well on common software development tasks, all outdistancing even the AT.□

Next issue: 286 "Short Cards" that plug into the 8088 socket: a look at Victor's SpeedPac 286, Orchid's TinyTurbo 286, and a few other contenders.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



LANGUAGES

Turbo Prolog

Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066
(408) 438-8400
TELEX 172373

For those of you who haven't read about Turbo Prolog elsewhere: Borland has announced the language for the PC, to be released in April.

Borland claims that Turbo Prolog outperforms other Prolog implementations by a factor of up to 10,000, with performance comparable to prototypes for the Japanese 5th generation.

Borland has this to say about Prolog: " . . . so named because it is designed for Programming in Logic. . . .With Prolog, programmers need only describe facts, and let the computer determine the solution on its own. . . . Prolog brings the capability to infer or derive information from stated facts.

"The Prolog language . . . utilizes pattern-matching and back tracking, and this is a true part of AI technology."

Turbo Prolog's incremental compiler generates native code, linkable object modules, and a linking format compatible with the PC DOS linker. The product also supports an object-oriented Type System.

The language will include a full screen text editor that positions the cursor in the source code at the point of an error. Turbo Prolog programs will call the editor, and you can view and modify the running program's source code at run-time.

While working, the programmer will be able to move between a number of windows: for the editor, the debugging facility, the source code listing, and a sample query. The program also includes a help system and pull down menus.

Turbo Prolog has no limitation on the size of compiled modules or module source code. Include files are supported.

Modules can be linked with other programming languages using a standard Linker.

Type checking can be performed across module boundaries. Library modules allow the system to generate stand-alone files.

The program will have a full I/O library, including formatted I/O, streams and random access files, and debugging facilities that include deterministic analysis. The program will support color, graphics, turtle graphics, sound, and windowing.

The price? *$895.00*? No, *$99.95*.

Philipe Kahn just several months ago called AI "a bunch of manure," too.

Among Prolog, Lisp, Smalltalk, and Forth, Prolog has been creeping into first place around *PJ* as the language with the favorite syntax.

Assembler, of course, can do anything that AI can do. But Prolog's an interesting tool—maybe more interesting than putting multiple applications in an 80 x 25 display or putting 32 users on a CPU.

MasterFORTH
Target Application Generation System (TAGS)
$495.00

Micromotion
8726 S. Sepulveda Blvd. #A171
Los Angeles, CA 90045

MasterFORTH is a direct-threaded language that uses a cache technique. Included in the MasterFORTH package is a complete assembler, screen editor, debugger with a decompiler, single-step trace and breakpoint facility, and a file interface that supports the primitives in Kernighan and Plauger's *Software Tools*. MasterFORTH includes a string package. It matches the implementation in *Mastering Forth* by Anderson and Tracy and is compatible with the Forth-83 standard.

The package includes a "target compiler" that generates optimized application programs. All unused commands in the Forth dictionary are stripped away, and programs are generated without headers. Programs for one computer can be created for another. The target compiler can generate RAM-based or ROM-based applications, and can be modified by a set of SAFE specification commands. Floating point coprocessor 8087/80287 support is also included.

BetterBASIC Version 2.0

interpreter	***$199.00***
run-time system	***$250.00***
math chip support	***$ 99.00***
sample disk	***$ 10.00***

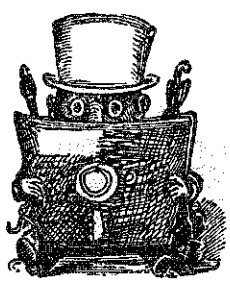
not copy protected

Summit Software Technology
106 Access Rd.
Norwood, MA 02062
(617) 769-7966

This is the popular BASIC, the previous version of which provides incremental compilation, procedures and functions, windows, and full DOS memory usage.

The new release can be used with existing BASICA and Microsoft GW-BASIC programs. Other additions:

1. Interactively change the type of numeric variables
2. Delete any unused variable or procedures/functions
3. Set BREAKPOINTS in and program and CONTINUE
4. Trace the execution of a program
5. Use GOTO out of but not into block structures
6. Use ELSE and ELSE IF clauses in DO IF/END DO block structures
7. Use the new GOTO END instruction
8. Locate arrays and records inside other arrays or records



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DESMET C EXTRAS

STRAIGHT FROM OUR CAST OF THOUSANDS

WELL...AT LEAST TEN OR SO.

BBS-1: Programs from our BBS incl. ARC, WINDOW, EXE2COM.C, DOSEDIT, EXE2ROM & more **$8**

BBS-2: ARC, O88 (C88 optimizer), CALENDAR, & more **$8**

D88: source level debugger **$50**

DOS LINK OPT: converts .O format files to .OBJ format **$35**

GRAPHICS: high/med/low res graphics (with source) **$35**

HACKER: source for _CSETUP, DOS 2.0 I/O, RAM.COM, a program to dump .O files & a little more **$25**

SEEtm : full screen editor (provided free with compiler) **$50**

MAKE: full UNIX-level MAKE. Recreates any system from its specified sources. Full macro & path name support. This product is not DeSmet specific **$50**

TOOLS: Tools and Utilities that DeSmet and Ouye use. (with Source) **$35**

XARRAY: a DOS eXtended ARRAY pkg which lets you define and access data arrays in the RAM above your C address space. Arrays any size to avail. memory. (With Source) **$39**

DESMET C FOR THE MACINTOSH: full K&R C w/ mach. level debugger. **$150**

DESMET C FOR THE PC — compiler, binder, assembler, std. and 8087 libs., SEE[tm] (full screen editor), Clist, Librarian, Compare and Profile **$109**

THE C PROGRAMMING LANGUAGE by Kernighan & Ritchie **$25**

C WARE
CORPORATION

505 W. OLIVE, SUITE 767
SUNNYVALE, CA 94086 USA
(408) 720-9696 TELEX: 358185
Call, write or telex to order by
VISA, MC or American Express

True BASIC version 1.1 with Hercules Support
$189.00, $39.95 as an upgrade

True BASIC, Inc.
39 South Main St.
Hanover, NH 03755
(603) 643-3882

True BASIC has released a new version of the True BASIC language system to support the Hercules monochrome graphics card. According to TBI, True BASIC is the first language that allows Hercules graphics users to access 640K and the 8087 processor.

According to the company, True BASIC's graphics syntax is hardware-independent, similar to the GKS graphics standard. This means that graphics developed using the Hercules card will be directly portable to systems supported by True BASIC's other adaptations —for the IBM Color Graphics Adapter, the Macintosh, and the Amiga.

This latest version of True BASIC's graphics syntax supports user-defined coordinates, open multiple windows, and animation. Built-in 2-D transforms give the user the ability to rotate, shift, shear, or rescale an image. Users can develop 720 x 348 graphics applications on the IBM monochrome display.

TBI used Digital Research's GEM VDI package for development and will ship a Hercules driver with each copy under a license agreement between TBI and DRI.

MEMORIES

Eccel AT Multifunction Board
$595.00

Orchid Technology
47790 Westinghouse Dr.
Fremont, CA 94539
(415) 490-8586

Bill Berkman of Orchid Technology called to talk about Orchid's Eccel AT multifunction board. It comes with a serial and parallel port or dual parallel ports and space for up to 3 Megs of memory. You can use up to 4 Eccels at once.

The board features mainframe-style ECC, or error correction code, which detects and corrects parity errors before the machine locks up. According to Bill, with memories of megabyte size, you are apt to get a parity error as often as once every two weeks.

The board can be used for DOS, Extended, or EMS memory simultaneously and comes with RAM Disk, Disk Caching, and Print Spooling software.

It runs at up to 6MHz with no wait states. If you are running an AT or clone at 8MHz or above, then one wait state needs to be inserted.

OTHER PRODUCTS

Logic Line
Logic-Line 1 $250.00
Logic-Line 2 $400.00

Thunderstone Corporation
(formerly Clarity Software)
P.O. Box 839
Chesterland, OH 44026

Logic-Line 1 came out in the fall, fast searching software built along AI lines. The computer searches and leaves the separation of "hits" from "misses" to the user. You can use words, keywords, groups of words, and groups of keywords to search a file, data base, or groups of files or data bases. In addition to straight string searches, you can locate possible logical unions of information or inference.

The documentation is a little difficult, possibly because the writers are dealing with an area a bit outside the realm of standard computer software, and it's not been clear whether Clarity Software could really find a place with this product.

Now it appears that they have been finding a market. A press release announces a change of name for the company (to *Thunderstone Corporation*) and the release of a second product, Logic-Line 2, which adds the ability to search numerical data such as spreadsheets, chemical formulae, codes, and catalogued tax laws and to cross-correlate inference with text and numerics. The press release includes some program documentation updates, and conveys a sense that some users out there are starting to speak the same language as the company.

Mastering Turbo Pascal by Tom Swan, 450 pp.
$18.95, ISBN 6287-7

Hayden Book Company
10 Mullholland Dr.
Hasbrouck Heights, NJ 07604

Hayden has released frequent contributor Tom Swan's book on Turbo Pascal, designed to be of help to the novice as well as the pro. *Mastering Turbo Pascal* contains 130 program examples and includes such advanced topics as writing interrupt routines; installing device drivers; displaying disk directories; writing large programs using overlays, chaining, and direct program execution; and calling DOS routines. It looks like a thorough job.□

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ BILLBOARD

PJ Billboard *is a product and services directory containing listings for hardware and software of interest to the IBM PC programmer. The goal is to provide our readers with a wide range of information on current products and updates.* ***PJ Billboard*** *serves our advertisers by providing a low cost, flexible format in which they can present their products to our readers.*

The standard layout is four columns, 9 picas wide, with copy set in 9 point on 10 point leading. ***PJ Billboard*** *offers a 400-character product description, including your company name, address, and telephone number. Boldface and italics are available.*

The listings are $65, prepaid, non-commissionable for a one-time insertion. Listings must be received typed, preferably on one of our convenient order forms. Copy deadlines are the same as the normal ad materials closing dates, which can be found on our rate card, or you may call the ***Programmer's Journal*** *Advertising Department at 503-484-0602 for more information.*

There were those who judged Melvin by his peripherals.. some even charged that he had character device errors.

ROMable CODE on PC!
PCLOCATE allows PC users to develop ROM-based software from MS-DOS "Exe" files. The user specifies the physical location of all segments. Output files are compatible with most PROM programmers. PCLOCATE supports the 8086, 8088, 80186, 80188, and 80286 processors. MC/VISA.

ALDIA SYSTEMS INCORPORATED
P.O. Box 37634
Phoenix, AZ 85069
(602) 866-1786

SMARTWARE International, Inc.,
introduces:

- DISK MANAGEMENT FACILITY (DMF) including
- FILE SUPPORT FACILITY (FSF)
- Cosmetically appealing display of up to 24 files at a time;
- Single character entries to perform the following:

-Erase multiple files (up to 24 at a time);
-Unerase multiple files (up to 24 at a time) previously erased with this facility;
-Erase a complete directory of files and/or disk;
-Change file attributes;
-Type ASCII files to the screen;
-Select a directory for display;
-Browse files with update capability—view ASCII & HEX data (including hidden & system files);
-Dump viewed files to print w/range selection in an easy to read functional format;
-Run .EXE, .COM & .BAT files;
-Execute DOS commands;
-Rename, move or copy up to 14 files at a time, placing each in a different subdirectory or drive if desired.

- An extremely functional tool for users and technical personnel alike.
- A true value $49.95 (for both facilities) or priced separately at $39.95 each. MC/VISA

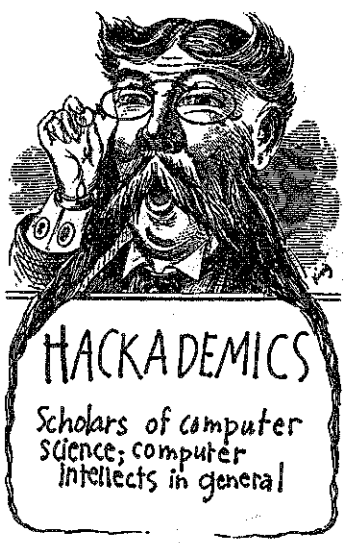
SMARTWARE International, Inc.
759 Avery Street
South Windsor, CT 06074
(203)646-2816

Documentation

- Writing, editing, formatting for photocopying or typesetting
- MS DOS, Apple, AT&T 3B series UNIX
- Over thirty manuals
- Credits include
  Dave Winfield's BATTER UP!
  (Billboard Top Twenty)
  Action Inn Management
  (AT&T certified)
  SmartCAM
  (MS DOS, Apple)
  Programmer's Journal
  (Managing Editor)

Erin O'Connor
c/o Programmer's Journal
P.O. Box 30160
Eugene, OR 97403
(503) 484-2162

WIRELESS KEYBOARD
CHERRY KXN3-8451
Operates up to 40 ft. from terminal, IBM PC, PC jr., and PC/XT compatable. Made in the USA, proven Cherry key module. Infared receiver included. Limited quantity. $149.00
Saratoga Electronics
408-446-4949



King Computer Vocabulary Entry #1

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ERRATA

There were two typographical errors in my article "Funky String Functions" (*PJ*, March/April 1986). On page 24, the first program listing in the left column starts

Word(s : OneString) : OneString;

but should have read:

***PROGRAM* Word(s : OneString) : OneString;**

Later on that same page, near the top of the right column, the line

TestToken;

should have read:

***PROGRAM* TestToken;**

Thanks for bringing these errors to the attention of *PJ* readers.

Tom Swan
Swan Software
P.O. Box 206
Lititz, PA 17543
(717) 627-1911

You're welcome, Tom. We're rounding up the usual suspects. □

FORTRAN PROGRAMMERS

Downloading from mainframes or developing on the PC the choice is F77L — Lahey Fortran.

"Lahey's F77L FORTRAN is the compiler of choice... F77L compiled the five files in a total of 12 minutes which was 4 times as fast as MS FORTRAN and an astounding 6 times as fast as Pro FORTRAN." PC Magazine

"The manual that comes with this compiler is well put together. The messages are clearly explained, the compiler's unique features are well documented... All in all, F77L is a fine, well supported product that we think will do very well in the marketplace."
Computer Language

VERSION 2.0 NOW AVAILABLE — $477

Full ANSI FORTRAN-77
Source On-Line Debugger
Extensions for easy mainframe porting
Common/Array greater than 64K
Lattice C and other 3rd Party Compatibility

To order or for more information

(213) 541-1200

Lahey Computer Systems, Inc.
31244 Palos Verdes Drive West, Suite #243
Rancho Palos Verdes, CA 90274

Requires MS-DOS and 8087

MS-DOS and MS FORTRAN are trademarks of Microsoft Corporation
Pro FORTRAN is a trademark of International Business Machines

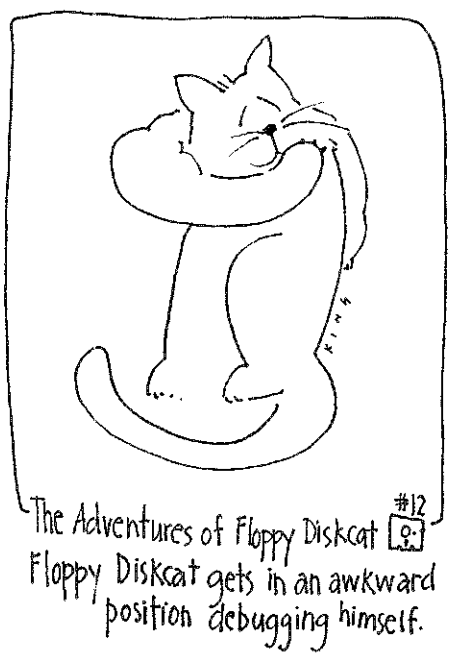
Circle 44 on the reader service card

ADVERTISER INDEX

	Circle Number	Page Number
American Alternative Energy	35	34
Applied Reasoning	50	50
ATRON	1	12
Austin Code Works	30	45
BC Associates	36	33
Blaise Computing	2	26
Borland International	3	11
Brady Books	37	17
C-Ware	38	60
Compuview Products, Inc.	39	41
Creative Prog. Consultants	40	40
Dair Computer Systems	5	56
Data Base Decisions	6	13
Datalight	7	43
Design Solutions	41	57
Ecosoft, Inc.	8	Cov. 4
Entelekon	9	19
Gimpel Software	11	25
Harvard Softworks	42	10
Interface Technologies	43	7
Lahey Computer Systems	44	62
Landmark Software	12	28
Martin Scott	27	16
MetaWare	45	Cov. 3
Micromotion	13	3
Oliver Computing	15	24
Opt-Tech Data Processing	16	29
Peter Norton Computing	17	23
Phoenix Computer Products	18	15
Phoenix Computer Products	51	55
Quantam Information Systems	31	58
Quilt Computing	46	46
Rational Systems	20	22
Soft Advances	33	46
Softcraft	21	Cov. 2
Software Masters	32	5
Solution Systems	47	14
Syscom Software	34	46
Visual Age	48	35
Wizard Systems Software	49	44
Zedcor, Inc.	28	20

PUT IN A GOOD WORD

When you contact these Advertisers for information, please be sure to mention that you saw their product advertised in **Programmer's Journal**. Thanks!



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PEER REVIEW

Volume 4, Number 3

Rank the articles in this issue of *Programmer's Journal.*

____ PJ FORUM: FNE, DTACK GROUNDED
____ REDIRECTIONS: Frank D. Greco, Sun Microsystems
____ CONSULTING: Paul Barkley, Reusable Modules
____ PASCAL: Bela Lubkin, Critical Error Handling for Turbo
____ ASM: George Defenbaugh, Memory Model and Language Independence
____ ASM: C.D. Hawkinson, Approach to Windowing
____ C: Rex Jaeschke, The Proposed ANSI C Standard, Part 2
____ C: William Hunt, Microsoft and Lattice 3.0
____ REVIEW: Accelerator Boards for Software Development

NAME _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

TELEPHONE _____

For more information on the products advertised in *Programmer's Journal*, just fill out this postage paid self mailer and drop it in the mail. Our Advertisers will be glad to send you FREE information on the products you are interested in.

Circle the number(s) below that correspond to the Reader Service number of the Advertiser from whom you would like information. Be sure to clearly print or type your name and address on the form at left.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31			
32	33	34	35	36	37	38	39	40	41	42	43	44	45			
46	47	48	49	50	51	52	53	54	55	56	57	58	59			
60	61	62	63	64	65	66	67	68	69	70	71	72	73			
74	75	76	77	78	79	80	81	82	83	84	85	86	87			
88	89	90	91	92	93	94	95	96	97	98	99	100				

PLEASE RESPOND BY: AUGUST 31, 1986

READER INFORMATION CARD IV, 3

READER SERVICE CARD

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 44 EUGENE, OR

POSTAGE WILL BE PAID BY ADDRESSEE

programmer's journal
The Resource Journal for IBM PC Programmers

P.O. BOX 30160
EUGENE, OR 97403

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BACK ISSUES

• Vol. 1, No. 1: Norton batch files and Pascal I/O; Program style manual; COBOL on the IBM PC; Benchmarks; System analysis; BASIC modules; Rounding error.

• Vol. 1, No. 2: Defining character sets; C86 BIOS calls; Troubleshooting; Copy protection; Pascal code; Input routines; Writing an assembler.

• Vol. 1, No. 3: ASM dumb terminal; ASM tutorial routines; C source code; ASM graphics line drawing; Text editors reviewed; Utilities; Self-documenting programs; Text files to dBASE II; dBASE II PEEKS & POKES.

• Vol. 1, No. 4: Diskette directory via Pascal; MASM WHILEWEND; C/ASM interface; ASM stay resident routine; Structured programming; Software law; Pascal tutorial; Default values and variable names.

• Vol. 1, No. 5: Multi-tasking software review; Transporting C programs; Programmer's calculator; dBASE II bargraphs; Utilities; ROM BIOS equates; Pascal check program; Combining modules in assembler; Structured programming in BASIC; Agreement outline.

• VOL. 1, No. 6: ASM printer configuration; DOS patch utility; Hi-speed graphics; Software ownership; Structured programming revisited; C and BASIC compared; Software protection; dWINDOW review.

• Vol. 2, No. 1: Freelance consulting; Sorting in C; Pascal text editor source code; CRT distress; Compiled BASIC; Assembly modules for C; Designing a relational database; Ultra Utilities; Punctuation and Style reviewed.

• Vol. 2, No. 2: Interview with Brian Kernighan; Understanding C through assembly language; Utility routines in C and assembly; Music in C; Automatic line number optimizing in BASIC; Assembly language tutorial; Full-screen input routines in BASIC; dBASE II printer control codes; Pascal tools reviewed; C programming support products.

• Vol. 2, No. 3: Software development environments; Creating a productive screen handling tool; Assembly tutorial; Generating graphics within windows; Composition and editing system for musical ideas; Hybridizing assembly and BASIC; Creating screens; Linking Assembler to Pascal and BASIC.

• Vol. 2, No. 4: Writing a custom ROM BIOS for PC DOS; RISC, transputers, and mass storage; The future of disk protection; 8088 assembly macros; Random access files in BASIC; C utility routines you can buy; using the PMATE macro language; Choosing a database manager; E/SP syntax-checking; Pascal editor reviewed.

• Vol. 2, No. 5: Developing the APC CORE Executive; From Assembler to C via a 211 byte .EXE main(); Scroll control using BASIC and Assembler; Faster screen handling with BASIC and ASM, Random access files (Part two); RTOOL, a PC-resident programmers toolkit; dBkey reviewed; Pascal's pointer power.

• Vol. 2, No. 6: Periscope—building a hardware board for the AIBM PC; Intel 80286 bugs by FNE; AT&T surprises all; Field usage and pointer subtraction in C; Calendar functions for BASIC; Memory segmentation; Reduce your .EXE file sizes; Enhancing Debug; dBASE II to dBASE III? THINKTANK reviewed.

• Vol. 3, No. 1: November's Comdex; C—its uses and abuses; Using the C preprocessor; Memory segmentation examples; Index files (Part I); Structured programming language; Periscope, a symbolic debugger.

• Vol. 3, No. 2: UNIX! (again); Artificial intelligence; Doctor C's tips—pointers to functions; Calendar date routines in BASIC, Calculate record layouts, automatically!

• Vol. 3, No. 3: NAPLPS and the professional programmer; Instant-C; Printer control from a batch file; Repositioning the stars in Starfinder; Reducing EXE file size through data positioning tools for better programming.

• Vol. 3, No. 4: More UNIX stuff; C programming books (Part One); C code excerpts (Part One); How to cure the 'dreaded locked keyboard' condition; Calling MS DOS from Turbo Pascal; DOS 2.x sort and find filters; the SET Tools.

• Vol. 3, No. 5: Windows done here; C programming books (Part Two); C code excerpts (Part Two); A menu-driven PC using DISPLAY.COM; Turbo Pascal and MASM: making them work together; dBASE II routines; Turbo Pascal 3.0; Top-View Programmer's Toolkit.

• Vol. 3, No. 6: About C's HLL overhead; How to recover all your costs; Report on the COMDEX and Usenix shows; The Santa Cruz Operation Xenix for the IBM PC; Common C bugs; CTOOLS2; Turbo Pascal 3.0 and the subdirectories; More dBASE II routines; Clipper reviewed; A better way to interface BASIC and Assembler.

• Vol. 4, No. 1: PC/AT hard disk problems and plunging 64K DRAM prices; AUTOINC explained; Sorting using an AVL tree; Combining binary trees; Using DOS to call programs from Turbo Pascal; An operating system for the people; Final installment of Brett Salter's dTOOLS; Two microcomputer graphics books reviewed.

• Vol. 4, No. 2: Intel 80386, single- vs. multi-user, multi-tasking computing, benchmarks; The home office syndrome; Industry news; Funky string functions; Capturing file handles in Turbo Pascal; Proposed ANSI C Standard, part 1; Optimizing for speed in assembler; Memory resident control over print screen functions; Structuring BASIC output with field statements; QuickBASIC Compiler and networking; Programming for printer output reviewed.

Yes! I want to start receiving programmer's journal

The Resource Journal for IBM PC Programmers

☐ Check enclosed Total amount _____

☐ Charge my Bankcard: ☐ VISA ☐ MC

_____ Expir. Date _____

_____ Signature

Back Issues are $5.50 ea. PREPAID ONLY

($6.50 Canada, $8.50 foreign, U.S. FUNDS ONLY)

Vol. I # __ __ __ __ __ __

Vol. II # __ __ __ __ __ __

Vol. III # __ __ __ __ __ __

Vol. IV # __ __ __ __ __ __

Bi-Monthly Subscription Rates

☐ New Subscription ☐ Renewal

Term	U.S.	Canada (U.S. funds only please)	Foreign (U.S. funds only please)
1 year, 6 issues	☐ $24	☐ $30	☐ $40
2 years, 12 issues	☐ $44	☐ $55	☐ $75
3 years, 18 issues	☐ $64	☐ $80	☐ $110

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C spoken here...

High C™

Do you want to use a C compiler that

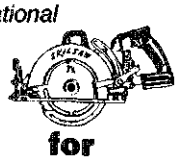
- was chosen by Ashton-Tate for implementing dBASE III® Plus
- was well rated in ***Computer Language, Feb. 86***
- "would have saved me three weeks of porting time had I had High C instead of Microsoft's new C"
  Mike LeBlanc, compiler developer, Sky Computers
- "is the *only* C compiler for the IBM PC capable of compiling NYU's Ada/Ed compiler"
  Dave Shields, research scientist, New York Univ.
- has a complete run-time library
- has structure assignment, **enum**, **void**...
- supports nested functions as in Pascal
- supports pcc and full K&R C *plus* some latest, nifty extensions from the new ANSI-proposed C standard
- "saved 15% of code over five large modules of MultiMate relative to Lattice C"
  David Beauchesne, Multimate International

Pascal spoken here...

Professional Pascal™

Do you want to use a Pascal compiler that

- was chosen by Lifetree Software, Inc., for implementing Volkswriter Deluxe™
- was well rated in ***Computer Language, May 86***
- serves as a systems and applications language at CAD/CAM giant Daisy Systems Corporation
- has 8-, 16-, and 32-bit integers; sets up to 64K bits
- has varying-strings of up to 64K characters
- has a full-fledged C macro preprocessor
- has many run-time library additions: UNIX™-like I/O, multiple heaps, interrupts, . . .
- has all the bit-pushing operators of C
- has *many* more extensions, getting you half way to Ada® for a non-Ada price
- is the "howitzer" of Pascals and "could well be the most powerful Pascal compiler ever implemented on a microcomputer"
  PC Magazine, October 29, 1985, p. 144



Power Tools for Power Users

Each compiler • generates **superb code**, with optimizations such as common-subexpression elimination and cross-jumping • sports no less than ***five* memory models** for the 8086 (Small, Compact, Medium, Big, and Large) • supports a unique implementation of register variables • supports the **8087/80287** in native mode, or **emulates** • supplies *three* floating-point formats • generates special instructions for the 80186/286 • generates code that runs in **80286 protected mode** • gives you **hundreds of error and warning messages**, helping you find those subtle bugs *before* you need a debugger's help • lets you **overlay *data*** as well as code (when used with PLINK86), for substantial space savings • lets you write **interrupt routines** directly in high-level language • lets you get "close to the machine" with built-in move/scan/compare operations • is supported by equivalent **resident and cross** compilers for the 80286 (Concurrent DOS 286), 68010 (UNIX 4.2, GEM-DOS™), 32032 (UNIX 4.2), VAX (UNIX 4.2, VMS), IBM 370,... • contains a **multi-modular cross-referencer** • produces ROM-able code for embedded applications • can talk to those **other languages** by those other vendors • gives you **64K run-time stack** space that can be shared with the heap • is endowed with an **amazing number of pragmas** (compiler controls) for customization to your application • has a compiler start-up **profile** • was designed for professional software developers, not hobbyists • comes with great technical support • comes with extensive **typeset documentation** • and *more*... call or write for your information packet today...

Not recommended for casual use, but for applications needing **industrial-strength tools**, contact

MetaWare™
INCORPORATED
412 Liberty Street, Santa Cruz, CA 95060, (408) 429-6382 (429-META)

High C: $495 —on any MS/PC-DOS system— **Professional Pascal:** $595

OEMs: Contact us about porting our professional compilers to your systems.

TWS: Professional Compiler Developers and competitors, ask about our Translator Writing System compiler toolbox; see the review in ***Computer Language,* December, 1985.**

DOS Helper™: Need UNIX-like utilities to enhance MS-DOS? Ask about our powerful tools for $49.95—included free with MetaWare compilers: FIND, TAIL, MV, LS, CAT, UNIQ, FGREP, and WC.

Circle 45 on the reader service card

• MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated • Other trademarks and their owners are: UNIX—AT&T, dBASE III—Ashton-Tate, Volkswriter Deluxe—Lifetree Software, GEM-DOS—Digital Research, Ada-DoD. © 1986 MetaWare.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NEW RELEASE

Ecosoft's Eco-C88 Rel. 3.0 C Compiler

$59.95

Release 3.0 has new features at an unbelievably low price. ECO-C88 now has:

- Prototyping (the new type-checking enhancement)
- enum and void data types
- structure passing and assignment
- All operators and data types (except bit fields)
- A standard library with more than 200 functions (many of which are System V compatible for greater code portability)
- cc and mini-make that all but automates the compile process
- 8087 support (we sense the 8087 at runtime – no dual libraries)
- ASM or OBJ output for use with MSDOS linker
- Tiered error messages – enable-disable lint-like error checking
- Fast compiles and executing code
- Expanded user's manual
- Enhanced CED program editor (limited time offer)

We also offer the following support products for Eco-C88.

CED Program Editor — $29.95

CED now supports on-line function help.
If you've forgotten how to use a standard library function, just type in the name of the function and CED gives you a brief summary, including function arguments. CED is a full screen editor with auto-flagging of source code errors, multiple windows, macros, and is fully configurable to suit your needs. You can edit, compile, link, and execute DOS commands from within the editor. Perfect for use with Eco-C88. For IBM PC, AT and look alikes.

C Source for Standard Library — $10 ($20 if not with order)

Contains all of the source code for the library functions that are distributed with Eco-C88, excluding the transcendentals and functions written in assembler.

Developer's Library — $25 ($50 if not with order)

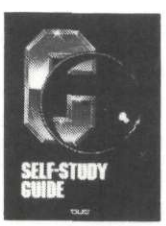
Contains the source code for all library functions, including the transcendentals and those written in assembler. Perfect for the developer that wish to write their own custom functions or learn how we implemented the Eco-C88 library.

ISAM Library — $15 ($30 if not with order)

Contains the code from the **C Programmer's Library** in relocatable format (i.e., .OBJ) including the delete code for the ISAM file handler.

C Programming Guide — $20

After reading the 1st edition, Jerry Pournelle (**BYTE** Magazine) said: "I recommend this book... Read it *before* trying to tackle Kernighan and Ritchie." The second editon expands this best seller and walks you through the C language in an easy-to-understand manner. Many of the error messages include references to this book making it a perfect companion to Eco-C88 for those just starting out with C.



C Self-Study Guide — $17

(Purdum, Que Corp.). Designed for those learning C on their own. The book is filled with questions-answers designed to illustrate many of the tips, traps, and techniques of the C language. Although written to complement the Guide, it may be used with any introductory text on C.

C Programmer's Library — $20

(Purdum, Leslie, Stegemoller, Que Corp.). This best seller is an intermediate text designed to teach you how to write library functions in a generalized fashion. The book covers many advanced C topics and contains many useful additions to your library including a complete ISAM file handler.

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later. Call today:

1-800-952-0472 (for orders)
or
1-317-255-6476 (tech. info.)

ECOSOFT INC.

Ecosoft, Inc.
6413 N. College Ave. • Indianapolis, IN 46220

AMERICAN EXPRESS · MasterCard · VISA

TRADEMARKS: ECO-C88, ECOSOFT

Circle Number **8** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED